

PCT/JP2004/006970

日 本 国 特 許 庁
JAPAN PATENT OFFICE

01.7.2004

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日
Date of Application: 2003年 6月 6日

出 願 番 号
Application Number: 特願2003-162928
[ST. 10/C]: [JP2003-162928]

REC'D 22 JUL 2004

WIPO

PCT

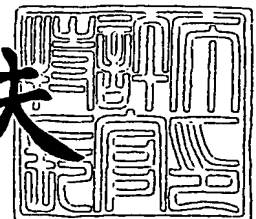
出 願 人
Applicant(s): シャープ株式会社

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN
COMPLIANCE WITH
RULE 17.1(a) OR (b)

2004年 5月12日

特許庁長官
Commissioner,
Japan Patent Office

今 井 康 夫



出証番号 出証特2004-3039577

【書類名】 特許願

【整理番号】 03J02117

【提出日】 平成15年 6月 6日

【あて先】 特許庁長官 殿

【国際特許分類】 G11B 27/036

【発明者】

 【住所又は居所】 大阪府大阪市阿倍野区長池町 2 2 番 2 2 号 シャープ株式会社内

 【氏名】 木田 歩

【発明者】

 【住所又は居所】 大阪府大阪市阿倍野区長池町 2 2 番 2 2 号 シャープ株式会社内

 【氏名】 木山 次郎

【発明者】

 【住所又は居所】 大阪府大阪市阿倍野区長池町 2 2 番 2 2 号 シャープ株式会社内

 【氏名】 山口 孝好

【特許出願人】

 【識別番号】 000005049

 【氏名又は名称】 シャープ株式会社

【代理人】

 【識別番号】 100080034

 【弁理士】

 【氏名又は名称】 原 謙三

 【電話番号】 06-6351-4384

【選任した代理人】

 【識別番号】 100113701

 【弁理士】

 【氏名又は名称】 木島 隆一

【選任した代理人】

【識別番号】 100116241

【弁理士】

【氏名又は名称】 金子 一郎

【手数料の表示】

【予納台帳番号】 003229

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 0208489

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 データ記録方法、データ記録装置、およびデータ記録媒体

【特許請求の範囲】

【請求項 1】

記録媒体に、A Vデータと、該A Vデータと同期するアフレコデータとを記録するデータ記録方法であって、

記録媒体上に、A Vデータの同一のシーンを使って複数のユーザプログラムを作成可能であり、かつ、アフレコデータが複数のユーザプログラムから参照可能であり、

ユーザプログラムとアフレコデータとの関係を管理する管理情報を上記記録媒体上に記録することを特徴とするデータ記録方法。

【請求項 2】

上記アフレコデータは、複数のユーザプログラムにおいて参照されるアフレコデータ群を含む領域中の部分領域であり、前記領域はひとつのアフレコファイルとして記録されており、

上記管理情報は、オリジナルプログラムに対して1つ存在する管理情報ファイル中にアフレコデータ毎に記録されることを特徴とする前記請求項 1 に記載のデータ記録方法。

【請求項 3】

上記アフレコデータは、同一のユーザプログラムにおいて参照されるアフレコデータ群ごとにアフレコファイルとして記録されており、

上記管理情報は、複数のオリジナルプログラムに対して1つ存在する管理情報ファイル中にアフレコファイル毎に記録されることを特徴とする前記請求項 1 に記載のデータ記録方法。

【請求項 4】

上記アフレコデータは、同一のユーザプログラムにおいて参照されるアフレコデータ群ごとにアフレコファイルとして記録されており、

上記管理情報は、オリジナルプログラムに対して1つ存在する管理情報ファイル中にアフレコファイル毎に記録されることを特徴とする前記請求項 1 に記載の

データ記録方法。

【請求項 5】

上記管理情報は、アフレコデータが複数のユーザプログラムから参照されている場合における参照回数を含むことを特徴とする前記請求項 1 に記載のデータ記録方法。

【請求項 6】

上記管理情報は、アフレコデータを参照しているユーザプログラム名を含むことを特徴とする前記請求項 1 に記載のデータ記録方法。

【請求項 7】

上記管理情報は、ユーザプログラムによって参照されているアフレコデータの参照範囲の情報を含むことを特徴とする前記請求項 1 に記載のデータ記録方法。

【請求項 8】

記録媒体に、A V データと、該 A V データと同期するアフレコデータとを記録するデータ記録装置であって、

記録媒体上に、A V データの同一のシーンを使って複数のユーザプログラムを作成可能であり、かつ、アフレコデータが複数のユーザプログラムから参照可能であり、

ユーザプログラムとアフレコデータとの参照関係を管理する管理情報を上記記録媒体上に記録することを特徴とするデータ記録装置。

【請求項 9】

A V データと、該 A V データと同期するアフレコデータとが記録されたデータ記録媒体であって、

記録媒体上に記録した A V データの同一のシーンを使って作成された複数のユーザプログラムが、複数のユーザプログラムによって同一のアフレコデータを参照可能とするように記録されており、

さらに、ユーザプログラムとアフレコデータとの参照関係を管理する管理情報が記録されていることを特徴とするデータ記録媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、映像データ、音声データをハードディスク、光ディスク等のランダムアクセス可能な記録媒体に対して記録・再生するデータ記録方法及びデータ記録装置に関するものである。

【0002】**【従来の技術】**

ディスクメディアや半導体媒体を用いたビデオや音声のデジタル記録再生装置が普及しつつある。それらにおいて、テープメディアと同様アフターレコーディング（アフレコ）機能を安価に実現する技術が求められている。アフレコ機能は、既に記録したオーディオやビデオに対し、後から情報、特にオーディオやグラフィックを追記する機能である。

【0003】

ここで、テープメディアにはないディスクメディアにおける特徴機能である、非破壊編集機能あるいはノンリニア編集機能と呼ばれる機能がある。すなわち、非破壊編集機能あるいはノンリニア編集機能では、ディスク上に記録したAVストリームを移動あるいはコピーすることなく、AVストリームの任意の区間を任意の順番で再生できる。この機能においては、単に再生順番を変えるだけでなく、同一のシーンを使って複数のユーザプログラムを作る、つまり同一のシーンを複数のユーザプログラムで共有することが可能となる。このときのユーザの要望としては、それぞれのユーザプログラムで個別にアフレコがしたい、というものがある。

【0004】

例えば、ユーザプログラムAではあるシーンに対してBGMをアフレコし、同じシーンを共有するユーザプログラムBではアナウンスをアフレコしたい、といった要望である。そのような要望に対して、特許文献1では、アフレコ用領域割り当てを管理する情報を備えることで、複数のユーザプログラムからアフレコした場合に、異なるユーザプログラム間で共有される同一のシーンのアフレコに対して、意図しない上書きをすることを防ぐことができる。

【0005】

上記特許文献1におけるディスクの記録フォーマットを図27に示す。ディスクにおける記録領域は、ECC (Error Correcting Code) ブロックの列で構成される。ECCブロックは符号化を行う際の最小単位であり、データに加えエラー補正用のパリティが付加され、符号化が行われている。ディスク内のデータを読み込む際は、このECCブロックの単位で読み込み誤り訂正をしてから、必要なデータを取り出す。

【0006】

一方、ディスクに対してデータの書き換えを行う際は、まずECCブロック単位で読み込み、誤り訂正をしたデータに対して必要な部分を書き換え、再度誤り符号を付与し、ディスクに記録を行う。

【0007】

ビデオ (映像) データやオーディオ (音声) データは、ECCブロック中で、アフレコデータブロック、オリジナルオーディオブロック、オリジナルビデオブロックの順に配置される。それぞれのブロックにはほぼ同じ時間に対応するアフレコデータ、オリジナルオーディオデータ、オリジナルビデオデータが含まれている。

【0008】

尚、オリジナルオーディオブロックとオリジナルビデオブロックとを合わせてオリジナルブロックと呼ぶことにする。オリジナルプログラム (アフレコデータを記録する前の映像) を記録する際は、アフレコデータブロックにダミーのデータを書き込んでおく。

【0009】

アフレコデータの記録の際は、アフレコデータブロックヘデータの書込みを行う。このとき、アフレコデータブロック中の記録を行った領域については既割り当て領域とし、残りを未割り当て領域としてオリジナルプログラム中の管理情報内で別々に管理する。この管理情報を参照することで、異なるユーザプログラム間でのアフレコデータの上書きを防止することができる。新たなアフレコデータが記録された際には、それぞれの領域における管理情報の書き換えを行う。

【0010】

【特許文献1】

特開 2002-373480号公報（公開日平成14年12月26日）

【0011】**【発明が解決しようとする課題】**

ところが、上記従来の構成では、ユーザプログラムを削除する場合を考えると、削除されるユーザプログラムに対応して不要となったアフレコデータを削除して領域を開放し、アフレコデータブロック領域を有効に再利用したいという要求に対しては、以下のような問題が生じる。

【0012】

すなわち、同一のシーンを複数のユーザプログラムで共有する場合、各アフレコデータは複数のユーザプログラムから参照されている可能性があり、これを安易に削除することはできない。これを、より具体的に説明すると以下のとおりである。

【0013】

つまり、あるユーザプログラムを削除するにあたって、この削除されるユーザプログラムで参照されているアフレコデータを併せて削除しようとする場合には、削除されるべきアフレコデータは削除されるユーザプログラムのみで参照されるものであり、他のユーザプログラムにおいて共有されていないことが必要である。

【0014】

しかしながら、従来では、ユーザプログラムの削除を行った場合、参照していたアフレコデータが他のどのユーザプログラムからも参照されておらず削除可能であると判断されるためには、すべてのユーザプログラムの管理情報をチェックする必要がある。このようなチェック動作は、シーク時間の長い光ディスクにおいては処理時間の面で問題となる。

【0015】

本発明は、上記の問題点を解決するためになされたもので、その目的は、ユーザプログラムの削除時、該ユーザプログラムにおいて参照していたアフレコデータが削除可能であるか否かの判断を容易にすることができるデータ記録方法および

びデータ記録装置を提供することにある。

【0016】

【課題を解決するための手段】

本発明のデータ記録方法は、上記の課題を解決するために、記録媒体に、AVデータと、該AVデータと同期するアフレコデータとを記録するデータ記録方法であって、記録媒体上に、AVデータの同一のシーンを使って複数のユーザプログラムを作成可能であり、かつ、アフレコデータが複数のユーザプログラムから参照可能であり、ユーザプログラムとアフレコデータとの関係を管理する管理情報を上記記録媒体上に記録することを特徴としている。

【0017】

上記の構成によれば、記録媒体に、AVデータと、該AVデータと同期するアフレコデータと記録するにあたって、記録媒体上に、AVデータの同一のシーンを使って複数のユーザプログラムを作成可能であり、かつ、アフレコデータが複数のユーザプログラムから参照可能となっている。

【0018】

このため、例えば、あるユーザプログラムを削除するにあたって、この削除されるユーザプログラムで参照されているアフレコデータを併せて削除しようとする場合には、削除されるべきアフレコデータは削除されるユーザプログラムのみで参照されるものであることを確認することが必要となるが、このような確認が上記管理情報によって容易に行える。

【0019】

また、上記データ記録方法では、上記アフレコデータは、複数のユーザプログラムにおいて参照されるアフレコデータ群を含む領域中の部分領域であり、前記領域はひとつのアフレコファイルとして記録されており、上記管理情報は、オリジナルプログラムに対して1つ存在する管理情報ファイル中にアフレコデータ毎に記録される構成とすることができる。

【0020】

また、上記データ記録方法では、上記アフレコデータは、同一のユーザプログラムにおいて参照されるアフレコデータ群ごとにアフレコファイルとして記録さ

れており、上記管理情報は、複数のオリジナルプログラムに対して1つ存在する管理情報ファイル中にアフレコファイル毎に記録される構成とすることができる。

【0021】

上記の構成によれば、アフレコファイルのサイズとアドレスとが常に変わらないような管理が可能であり、アフレコデータの削除が管理情報の書き換えのみで行うことができるため、ファイルシステムの書き換えを必要としない。

【0022】

また、上記データ記録方法では、上記アフレコデータは、同一のユーザプログラムにおいて参照されるアフレコデータ群ごとにアフレコファイルとして記録されており、上記管理情報は、オリジナルプログラムに対して1つ存在する管理情報ファイル中にアフレコファイル毎に記録される構成とすることができる。

【0023】

上記の構成によれば、アフレコデータ毎にアフレコファイルが分かれているため、アフレコデータを特定する際、データのオフセットなどから検索しなくとも、各アフレコファイルの参照ファイル名から即座に特定できるため、データの特定がより簡便に、より迅速に行うことができる。

【0024】

また、上記データ記録方法では、上記管理情報は、アフレコデータが複数のユーザプログラムから参照されている場合における参照回数を含む構成とすることができる。

【0025】

上記の構成によれば、例えば、あるユーザプログラムを削除するにあたって、この削除されるユーザプログラムで参照されているアフレコデータにおいて、該アフレコデータがユーザプログラムから参照されている参照回数が1減らされる。そして、この参照回数を減らす処理によって参照回数が0となったアフレコデータは、削除されるユーザプログラムのみで参照されるものであることが容易に確認できる。

【0026】

また、上記データ記録方法では、上記管理情報は、アフレコデータを参照しているユーザプログラム名を含む構成とすることができる。

【0027】

上記の構成によれば、あるユーザプログラムを削除するにあたって、この削除されるユーザプログラムで参照されているアフレコデータの削除が指示された場合に、該アフレコデータが他のユーザプログラムファイルから参照されているか否かの情報の他に、どのファイルから参照されているかをユーザに提示することができる。

【0028】

また、上記データ記録方法では、上記管理情報は、ユーザプログラムによって参照されているアフレコデータの参照範囲の情報を含む構成とすることができる。

【0029】

上記の構成によれば、各ユーザプログラムからアフレコデータのどの部分が参照されているかを示す参照範囲の情報を持たせることで、このような参照範囲情報により、アフレコデータ自体は複数のユーザプログラムから参照されていても要らなくなった部分のみを削除できるため、より有効に記録媒体の記録領域を活用することができる。

【0030】

【発明の実施の形態】

以下、本発明の実施形態について、図面を参照しながら詳細に説明する。ここでの説明は、本発明において共通に用いる構成、個々の実施形態に固有の内容という順に行っていく。

【0031】

<システム構成>

図2は、本実施の形態において共通に用いる、アフレコ可能なビデオディスクレコーダの構成である。

【0032】

このビデオディスクレコーダは、図2に示すように、バス100、ホストCP

U (Central Processing Unit) 101、RAM (Random Access Memory) 102、ROM (Read Only Memory) 103、ユーザインタフェース104、システムクロック105、光ディスク106、ピックアップ107、ECC (Error Correcting Code) デコーダ108、ECCエンコーダ109、再生用バッファ110、記録／アフレコ用バッファ111、デマルチプレクサ112、マルチプレクサ113、多重化用バッファ114、オーディオデコーダ115、ビデオデコーダ116、オーディオエンコーダ117、ビデオエンコーダ118および図示されないカメラ、マイク、スピーカ、ディスプレイ等で構成される。

【0033】

ホストCPU101は、バス100を通じてデマルチプレクサ112、マルチプレクサ113、ピックアップ107、また図示していないが、オーディオデコーダ115、ビデオデコーダ116、オーディオエンコーダ117、ビデオエンコーダ118との通信を行う。

【0034】

再生時には、ピックアップ107を通じて光ディスク106から読み出されたデータは、ECCデコーダ108によって誤り訂正され、再生用バッファ110に一旦蓄えられる。デマルチプレクサ112は、オーディオデコーダ115、ビデオデコーダ116からのデータ送信要求に従って、再生用バッファ110中のデータをその種別によって適切なデコーダに振り分ける。

【0035】

一方、記録時には、オーディオエンコーダ117とビデオエンコーダ118によって圧縮符号化されたデータは、マルチプレクサ113を介して多重化用バッファ114に一旦送られ、さらに、マルチプレクサ113によってAV多重化されて多重化用バッファ114から記録／アフレコ用バッファ111に送られる。記録／アフレコ用バッファ111中のデータは、ECCエンコーダ109によって誤り訂正符号を付加され、ピックアップ107を通じて光ディスク106に記録される。

【0036】

オーディオデータの符号化方式にはMPEG-1 Layer-IIを、ビデオデータの符号

化方式にはMPEG-2をそれぞれ用いることができる。

【0037】

光ディスク106は、外周から内周に向かって螺旋状に記録再生の行われるものであり、ビデオディスクレコーダに対して脱着可能な記録媒体とする。また、光ディスク106は、2048byteを1セクタとし、誤り訂正のため16セクタでECCブロックを構成する。ECCブロック中のデータを書き換える場合には、そのデータが含まれるECCブロック全体を読み込み、誤り訂正を行い、対象のデータを書き換え、再び誤り訂正符号を付加し、ECCブロックを構成して光ディスク106に記録する必要がある。また、光ディスク106では、記録効率を上げるためZCAV（ゾーン角速度一定）を採用しており、記録領域は回転数の異なる複数のゾーンで構成される。

【0038】

<ファイルシステム>

本実施の形態に係るビデオディスクレコーダでは、光ディスク106上の各種情報を管理するためにファイルシステムを用い、このファイルシステムをデータ（アフレコデータ、オーディオデータ、ビデオデータ）と共に光ディスク106に記録する。ファイルシステムには、パーソナルコンピュータ（以下、PCと略称する）との相互運用を考慮してUDF（Universal Disk Format）を使用することができる。ファイルシステム上では、各種管理情報やAVストリームはファイルとして扱われる。

【0039】

ファイルシステムのユーザエリアは、2048byteの論理ブロック（セクタと一対一に対応）で管理される。ユーザエリア上の各ファイルは、整数個のエクステント（連続した論理ブロック）で構成され、エクステント単位で分散して記録してもよい。ユーザエリア内の空き領域は、Space Bitmapを用いて論理ブロック単位で管理される。

【0040】

<ファイルフォーマット>

AVストリーム管理のためのフォーマットとして、ここではQuickTimeファイ

ルフォーマットを用いる。QuickTimeファイルフォーマットとは、Apple社が開発したマルチメディアデータ管理用フォーマットであり、PCの世界では広く用いられている。

【0041】

QuickTimeファイルフォーマットは、ビデオデータやオーディオデータ等(これらを総称してメディアデータとも呼ぶ)と管理情報とで構成される。メディアデータと管理情報とを合わせてここでは、QuickTimeムービー(略してムービー)と呼ぶ。メディアデータと管理情報とは、同じファイル中に存在しても、別々のファイルに存在してもよい。

【0042】

メディアデータと管理情報とが同じファイル中に存在する場合は、図3(a)に示すようなファイル構成をとる。各種情報はatomという共通の構造に格納される。ファイル201において、管理情報はMovie atomという構造に格納され、メディアデータのAVストリームはMovie data atomという構造に格納される。また、Movie data atomの最初には、Atom Header(AH)が配置される。尚、Movie atom中の管理情報には、メディアデータ中の任意の時間に対応するAVストリームのファイル中での相対位置を導くためのテーブルや、メディアデータの属性情報や、後述する外部参照情報等が含まれている。

【0043】

一方、管理情報とメディアデータとを別々のファイルに格納した場合は、図3(b)に示すような構成をとる。この場合、管理情報はファイル202においてMovie atomという構造に格納されるが、AVストリームはatomには格納される必要はなく、ファイル203に格納される。このとき、Movie atomはAVストリームを格納したファイル203を「外部参照」している、という。

【0044】

外部参照では、図3(c)に示すように、管理情報を格納する一つのファイル204によって複数のAVストリームファイル(この図ではファイル205・206)を参照することが可能である。この仕組みにより、AVストリーム自体を物理的に移動することなく、見かけ上編集を行ったように見せる、いわゆる「ノ

ンリニア編集」「非破壊編集」が可能になる。また、光ディスク106上に記録したAVストリームの同一のシーンを使って複数のユーザプログラムを作成することが可能であり、かつ、同一のアフレコデータを複数のユーザプログラムにて参照することも可能である。

【0045】

次に、図4ないし図14を用いて、QuickTimeの管理情報のフォーマットについて説明する。まず、共通の情報格納フォーマットであるatomについて説明する。atomの先頭には、そのatomのサイズであるAtom size、そのatomの種別情報であるTypeが必ず存在する。Typeは4文字で区別され、例えばMovie atomでは‘moov’、Movie data atomでは‘mdat’となっている。

【0046】

atom間には階層構造があり、各atomは別のatomを含むことができる。Movie atomの構成を図4に示す。Movie header atomは、そのMovie atomが管理するムービーの全体的な属性を管理する。Track atomは、そのムービーに含まれるビデオやオーディオ等のトラックに関する情報を格納する。User data atomは、ユーザが独自に定義可能なatomである。

【0047】

Track atomの構成を図5に示す。Track header atomはそのトラックの全体的な属性を管理する。Edit atomは、メディアデータのどの区間をムービーのどのタイミングで再生するかを管理する。Track reference atomは、別のトラックとの関係を管理する。Media atomは、実際のビデオやオーディオといったデータを管理する。

【0048】

Track header atomの構成を図6に示す。ここでは、後での説明に必要なもののみ説明する。Flagsは属性を示すフラグの集合である。代表的なものとして、Track enabledフラグがあり、このフラグが‘1’であればそのトラックは再生され、‘0’であれば再生されない。Track IDはトラック固有のIDである。Layerはそのトラックの空間的な優先度を表しており、画像を表示するトラックが複数あれば、Layerの値が小さいトラックほど画像が前面に表示される。

【0049】

Media atomの構成を図7に示す。Media header atomは、そのMedia atomの管理するメディアデータに関する全体的な属性等を管理する。Handler reference atomは、メディアデータをどのデコーダでデコードするかを示す情報を格納する。Media information atomは、ビデオやオーディオ等のメディア固有の属性情報を管理する。

【0050】

Media information atomの構成を図8に示す。Media information header atomは、ビデオやオーディオ等のメディア固有の属性情報を管理する。Handler reference atomは、Media atomの項で説明した通りである。Data information atomは、そのQuickTimeムービーが参照するメディアデータを含むファイルの名前を管理するatomであるData reference atomを含む。Sample table atomは、データのサイズや再生時間等を管理している。

【0051】

次にSample table atomについて説明するが、その前に、QuickTimeにおけるデータの管理方法について、図9を用いて説明する。QuickTimeでは、データの最小単位(例えばビデオフレーム)をサンプル(sample)と呼ぶ。個々のトラック毎に、サンプルには再生時間順に1から番号(サンプル番号)がついている。

【0052】

また、QuickTimeフォーマットでは、個々のサンプルの再生時間長およびデータサイズを管理している。また、同一トラックに属するサンプルが再生時間順にファイル中で連続的に配置された領域をチャンク(chunk)と呼ぶ。チャンクにも再生時間順に、1から番号がついている。すなわち、QuickTimeフォーマットでは、このようにチャンクが連続して配置されることにより一つのファイル901が形成される。

【0053】

さらに、QuickTimeフォーマットでは、個々のチャンクにおけるファイル先頭からのアドレスおよび個々のチャンクが含むサンプル数を管理している。これらの情報に基づき、任意の時間に対応するサンプルの位置を求めることが可能とな

っている。

【0054】

Sample table atomの構成を図10に示す。Sample description atomは、個々のチャンクのデータフォーマット(Data format)やサンプルが格納されているファイルのチャンクのインデックス等を管理する。さらに、Sample description atomの構成を図11(a)に示す。number-of-entriesは使用されているデータフォーマットの数を表し、その数だけSample description tableが存在する。図11(b)はオーディオのSample description tableの例である。

【0055】

図10のSample table atomにおいて、Time-to-sample atomは、個々のサンプルの再生時間を管理する。Sync sample atomは、個々のサンプルのうち、デコード開始可能なサンプルを管理する。Sample-to-chunk atomは、個々のチャンクに含まれるサンプル数を管理する。Sample size atomは、個々のサンプルのサイズを管理する。Chunk offset atomは、個々のチャンクのファイル先頭からのアドレスを管理する。

【0056】

Edit atomは、図12に示すように1個のEdit list atomを含む。Edit list atomはNumber of entriesで指定される個数分の、Track duration、Media time、Media rateの値の組(エントリ)を持つ。各エントリは、トラック上で連続的に再生される区間に対応し、そのトラック上での再生時間順に並んでいる。

【0057】

Track durationは、そのエントリが管理する区間のトラック上での再生時間、Media timeは、そのエントリが管理する区間の先頭に対応するメディアデータ上での位置、Media rateは、そのエントリが管理する区間の再生スピードを表す。尚、Media timeが‘-1’の場合は、そのエントリのTrack duration分、そのトラックでのサンプルの再生を停止する。この区間のことをempty editと呼ぶ。

【0058】

図13にEdit listの使用例を示す。ここでは、Edit list atomの内容が図13(a)に示す内容であり、さらにサンプルの構成が図13(b)に示す構成で

あるとする。尚、ここでは i 番目のエントリのTrack durationを $D(i)$ 、Media timeを $T(i)$ 、Media rateを $R(i)$ として示している。このとき、実際のサンプルの再生は、図13(c)に示す順に行われる。このことについて簡単に説明する。

【0059】

まず、エントリ#1は、Track durationが13000、Media timeが20000、Media rateが1であるため、そのトラックの先頭から13000の区間はサンプル中の時刻2000から33000の区間を再生する。次に、エントリ#2は、Track durationが5000、Media timeが-1であるため、トラック中の時刻13000から18000の区間、何も再生を行わない。そして、エントリ#3は、Track durationが10000、Media timeが0、Media rateが1であるため、トラック中の時刻18000から28000の区間において、サンプル中の時刻0から10000の区間を再生する。

【0060】

図14に、User data atomの構成を示す。このatomには、QuickTimeフォーマットで定義されてない独自の情報を任意個数格納することができる。1個の独自情報は1個のエントリで管理され、1個のエントリはAtom sizeとTypeとUser dataとで構成される。Atom sizeはそのエントリ自体のサイズを表し、Typeは独自情報をそれぞれ区別するための識別情報、User dataは実際のデータを表す。

【0061】

<インデックスファイル>

ディスク内に含まれるQuickTimeムービーを管理するため、インデックスファイルという特別なQuickTimeムービーファイルをディスク内に1個置く。

【0062】

インデックスファイルには、ディスク内のファイル(QuickTimeムービーやQuickTimeムービーから参照されている静止画等)に関するサムネイルや各種属性が登録されている。各種属性の中には、そのファイルが外部参照されている回数を示すlink countがある。

【0063】

link countを参照することで、そのファイルを参照しているファイルがあるかどうかを容易に知ることができ、他から参照されているファイルを不用意に削除

してしまうことを防ぐことができる。

【0064】

＜AVストリームの形態＞

AVストリームの形態を図15に示す。AVストリームは整数個のContinuous Unit (CU) で構成される。CUはディスク上で連続的に記録される単位である。CUの境界はECCブロック境界と一致するようにストリームが構成されている。またCUは、一つのPost Record Unit (PRU) と整数個のVideo Unit (VU) とで構成される。VUは独立に再生可能な単位であり、VUの集合が前述のオリジナルブロックに対応する。PRUは前述のアフレコデータブロックに対応する。PRUの領域サイズは、CUの再生時間にオーディオの最大ビットレートをかけたものである。

【0065】

＜ファイルシステムフォーマット＞

本発明の説明において用いるファイルシステムのフォーマットである、UDF (Universal Disk Format) について図16を用いて説明する。図16(a)に示すディレクトリ／ファイル構成をUDFで記録した例を図16(b)に示す。図中のAVDP602はAnchor Volume Descriptor Pointerの略であり、UDFの管理情報を探すためのエントリポイントに相当し、通常256セクタ目、Nセクタ目、あるいはN-256セクタ目 (Nは最大論理セクタ番号) に記録される。VDS601はVolume Descriptor Sequenceの略であり、UDFが管理する領域であるボリュームに関する管理情報を記録する。ボリュームは一般に一枚のディスクに1個存在し、その中にパーティションを一般に1個含む。FSD603はFile Set Descriptorの略であり、パーティションに1個存在する。パーティションの中での位置情報はパーティションの先頭からのセクタ番号に相当する論理ブロック番号で示される。なお、1個の論理ブロックは1セクタに対応する。また、各パーティションには図示しないがSpace Bitmapと呼ばれる各論理ブロックがファイルにすでに割り当てられているかそうでないかを示すテーブルが存在する。

【0066】

FSD603は、ルートディレクトリのFile Entry (FE) であるFE604の位置情報(論理ブロック番号と論理ブロック数で構成されextentと呼ばれる)を含む。FEは、extentの集合を管理しており、extentを書き換えたり、追加したり、削除することで、ファイルを構成する実データの順番を変えたり、データを挿入したり削除したりすることが可能である。FE604はルートディレクトリの直下のファイルやディレクトリの名称等を格納するFile Identifier Descriptor (FID) の集合を格納する領域605を管理する。領域605中のFID611、FID612は、それぞれファイル621、ファイル622のファイル名やextentの集合を管理するFE606、FE608の位置情報を含む。FE606はファイル621の実データを構成する領域である領域607、領域610をextentとして管理する。FE608はファイル622の実データを構成する領域である領域609をextentとして管理する。このときファイル621の実データにアクセスするためには、AVDP602、VDS601、FSD603、FE604、FID611、FE606、領域607、領域610の順にリンクを辿っていけばよい。

【0067】

〔実施の形態1〕

本発明の実施の形態1について、図1、図17ないし図24を用いて説明する。

【0068】

ここでは、アフレコに関する処理を中心に説明し、オリジナルプログラムのAVストリームの処理については特に説明しない。

【0069】

<初期記録時の処理>

PRUはオリジナルプログラムの記録の際にアフレコデータ用として確保される領域であり、AVストリームの記録時にアフレコファイルが作成されて、この領域はダミーデータとして管理される。

【0070】

オリジナルプログラムの管理情報内には、PRU領域内のアフレコデータが記

録されていない領域を管理するための未割り当てトラックを作成し、初期記録の際は、CU単位のPRU領域を1サンプルとしてすべてのPRU領域についての位置、サイズ、時間情報を管理するsample tableを作成する。この未割り当てトラックについては、Track header atomのTrack Enabled フラグを0にセットし、再生対象ではないことを示す。

【0071】

<アフレコデータ記録時の処理>

次に、オリジナルプログラムを外部参照したユーザプログラムを作成し、それにアフレコデータを記録する際の手順を説明する。ここでは、作成するユーザプログラムが、オリジナルプログラムの一部を抜き出して再生するというものである場合を考える。

【0072】

ユーザプログラムの作成は、オリジナルプログラムの管理情報からAVストリームのトラックのユーザが指定した区間のsample tableをコピーすればよい。

【0073】

このユーザプログラムにアフレコデータを追加するときの処理について図1に基づいて説明する。まず、ユーザプログラムのQuickTime管理情報をRAM102に読み込む。また、ユーザプログラムの管理情報のData reference atomに記述されたファイル名からオリジナルプログラムを特定し、参照先のオリジナルプログラムのQuickTime管理情報もRAM102に読み込む(S1501)。

【0074】

次に、ユーザが指定した区間におけるアフレコデータを記録すべきPRUの位置を、オリジナルプログラムの未割り当てトラックのsample tableから、アフレコ対象のビデオと時間的に対応する部分を探すことで特定する。そして、これから記録しようとするアフレコがオーディオの場合、ビットレートからデータ量を計算し、未割り当てトラックで管理しているPRUのsize内に収まるかどうかを判定する(S1502)。

【0075】

また、アフレコがグラフィックの場合は、記録するグラフィックデータのサイ

ズがPRUのsize内に収まるかどうかを判定する。具体的には、アフレコデータは、CU単位のスリーム再生時間分ずつ、ディスク上に均等に配置されているPRU領域に分けて記録されるため、ユーザが指定したアフレコ区間全てのPRU未割り当て領域のsizeを調べて判定を行う。

【0076】

アフレコデータを記録する領域が決定すれば、実際にその領域にアフレコデータを記録する。

【0077】

アフレコデータの記録が終了すると、次に、オリジナルプログラム側にアフレコしたデータを管理するトラックを新規に作成し(S1503)、記録したアフレコデータを管理するsample tableを作成する(S1504)。

【0078】

ここで、データの参照回数を管理するため、Track atom中のUser data atom('udta')内に、図17に示すような、独自のAtom typeであるreference management atom('rfmg')を定義する。このreference management atomの中に、データが参照されている回数を示すフィールド'ref#counter'を定義する。

【0079】

すなわち、Sample tableを作成した後、User data atom内に上記reference management atomを作成し、ref#counterフィールドに参照回数が1回であることを表す'1'をセットする(S1505)。

【0080】

次に、オリジナルプログラムの未割り当てトラックのsample tableのchunk offsetとsizeを、アフレコデータ記録後のPRUの残り領域に対応するよう更新する(S1506)。

【0081】

次に、記録したアフレコデータを再生するためのトラックをユーザプログラム側にも新規に作成し、先程作成したオリジナルプログラムと同様、アフレコトラックのsample tableを作成する。

【0082】

最後に、RAM102上のオリジナルプログラムおよびユーザプログラムのQuickTime管理情報を光ディスク106に記録する(S1507)。

【0083】

上記フローにおいて、S1502～S1506の処理は、図2に示す構成のビデオディスクレコーダでは、ホストCPU101が、RAM102内に読み込まれたオリジナルプログラムおよびユーザプログラムのQuickTime管理情報に対して行う。そして、S1502～S1506の処理が施されたオリジナルプログラムおよびユーザプログラムのQuickTime管理情報が、ホストCPU101に制御によってRAM102から読み出されバス100およびピックアップ107を介して光ディスク106に書き込まれる。

【0084】

一方、S1502で未割り当て領域がなかった場合、アフレコ用のビットレートを下げることで割り当て領域のサイズを小さくし、割り当て領域があるかどうかを再度調べるようにすることができる。それでも未割り当て領域がなければ、AVストリームのPRU領域とは別の領域に記録を行うことも可能である。アフレコする区間について、部分的に記録する領域がない場合、その区間のみPRU領域とは別の領域に記録するようにしても良い。

【0085】

アフレコデータ記録後のオリジナルプログラムは図18のようになる。アフレコが記録された区間のPRUは、アフレコデータが記録された領域(既割り当て領域)と未割り当て領域で構成される。オリジナルプログラムの管理情報であるMovie atomにはVideoトラック、Main Audioトラック、アフレコトラック(図中PR Audio)、未割り当てトラックの4つのトラックがあり、それぞれのデータのアドレスやサイズ、再生時間等の情報を持つ。

【0086】

既にアフレコデータが記録されている後に別の種類のアフレコデータを追加するときの処理も、新規にファイルを作成しないこと以外は同様である。図19(a)はアフレコデータ追加前のPRUの状態であり、図19(b)はアフレコデータ追加後のPRUの状態を示す。図19(b)では、既に記録されているアフ

レコ#Aトラックの後ろに、追加されるアフレコデータ用に新規にアフレコ#Bトラックを作成し、アフレコデータ毎に1トラックで管理を行う。Chunk offsetの値はPRU中の新規アフレコデータ開始位置をセットする。図では既にあったアフレコデータのすぐ後に新しいアフレコデータを記録しているが、未割り当て領域のどの位置でも記録を行って良い。新規データが追加される度に、残り領域に応じて未割り当てトラックのChunk offsetおよびSample sizeの更新を行う。

【0087】

また、ファイルシステム上では図20に示すように、オリジナルブロックで1ファイル（ストリームファイル）、アフレコデータブロックで1ファイル（アフレコファイル）の外部参照ファイルとして扱う。すなわち、アフレコファイル中に複数のアフレコデータが混在する形になる。

【0088】

アフレコファイル中に複数のアフレコデータが存在している場合でも、アフレコデータの種類毎にトラックを分け、トラック毎にref#counterを持つことで、容易に参照回数を管理することができる。

【0089】

＜ユーザプログラム作成時の処理＞

次に、アフレコデータを持つユーザプログラムをさらに非破壊編集する場合の処理を説明する。例えば図21のようにユーザプログラム1とユーザプログラム2をつなぎ合わせたユーザプログラム3を作成するときを考える。

【0090】

まず、ユーザプログラム1および2の管理情報をRAM102に読み込み、新たなユーザプログラム3として結合する。また、このユーザプログラム3のData reference atomに記述された参照ファイル名からオリジナルプログラムを特定し、オリジナルプログラムに関する管理情報をRAM102上に読み込む。

【0091】

次に、オリジナルプログラムの管理情報に関して、ユーザプログラム3のアフレコトラックのchunk offset情報から、offsetが同一のアフレコ済みサンプルをサーチすることで、複数あるアフレコデータの中からユーザプログラム3が参照

しているアフレコデータを特定し、それに対応するtrack atom内のudta atomのreference management atom ('rfmg')にあるred#counterを1増やす。

【0092】

最後に、RAM102上のオリジナルプログラムおよびユーザプログラム3の管理情報を光ディスク106に記録する。

【0093】

<ユーザプログラム削除時の処理>

ユーザから、ユーザプログラムの削除が指示された場合の処理について図22に沿って説明する。尚、はじめに削除指示されたユーザプログラムの管理情報をRAM102上に読み込んでおく。

【0094】

まず、ユーザプログラムの管理情報中のトラックを調べ、そのユーザプログラムが参照しているアフレコデータを管理しているオリジナルプログラムをリストアップする(S2101)。アフレコデータを管理するトラックがない場合は処理を終了する。

【0095】

次に、リストアップした各オリジナルプログラムに関して、以下の処理を行う。まず、オリジナルプログラムの管理情報をRAM102に読み込む。そして、ユーザプログラムのアフレコトラックのchunk offset情報からオリジナルプログラム内の対応するアフレコトラックを検索し、検索されたトラックのref#counterを1減らす(S2102)。

【0096】

ここでref#counterが0かどうかの判定を行い(S2103)、0であれば、削除を行ってこの部分の領域を開放し(S2104)、開放した領域を未割り当て領域に返却するように、未割り当てトラックのサンプル管理情報の更新を行う(S2105)。また、オリジナルプログラムの管理情報から削除を行ったアフレコデータに対応するTrack atomを削除する(S2106)。この処理はアフレコデータ記録時の処理の逆に対応する。

【0097】

最後に、ユーザプログラムを格納しているファイルを光ディスク106から削除(S2107)、オリジナルプログラムの管理情報を光ディスク106に記録する(S2108)。

【0098】

また、S2103において、ref#counterが0でない場合には、上記アフレコデータを参照している他のユーザプログラムを存在していることを示しているので、S2104～2106の処理、すなわちアフレコデータの削除処理を行わずにS2107に移行する。

【0099】

例えば、図23に示すような参照関係を持つファイルがあり、このうちユーザプログラム3を削除する場合を考える。ここでユーザプログラム3においては、アフレコデータAとアフレコデータBとが参照されている。ユーザプログラム3を削除するにあたって、アフレコデータAおよびアフレコデータBのref#counter数を1減らすと、図24に示すように、アフレコデータAのref#counter数は1になり、アフレコデータBのref#counter数は0になる。

【0100】

すなわち、アフレコデータBはユーザプログラム3にのみ参照されていたデータであるため、ユーザプログラム3と同時にアフレコデータBも削除されるが、アフレコデータAは他のユーザプログラムからも参照されている（ここでは、ユーザプログラム2から参照されている）ため削除されない。

【0101】

<実施の形態1における変形例>

本実施の形態1では、udta atom中に参照回数を示すフィールドを設けたが、本発明はそれに限定されるものではない。

【0102】

例えば、この参照回数を示す情報をSample description atom中のSample description table内に設けても良い。すなわち、アフレコデータ毎にSample descriptionを作成するという方法である。これにより、アフレコデータ毎にトラックを分割することを前提としなくても、Sample description atomで管理可能であ

るため、管理ファイルのデータ量の節約に繋がる。ただし、QuickTimeでは、一つのトラックに再生時間の重なるサンプルを同時に置くことはできないため、その場合はトラックを分割する必要がある。そうした場合でも、Sample description table内に参照回数を示す情報を持たせることで、より詳細にアフレコデータを管理することが可能になる。

【0103】

例えば、1つのアフレコデータの前半と後半とで異なるコーデックが使われており、Sample descriptionが分かれている場合などで、アフレコデータを記録したユーザプログラムを削除する場合、たとえ他のユーザプログラムからこのアフレコデータが参照されていたとしても、その参照範囲が前半部分のみであれば、後半部分のアフレコデータのみを削除して、そのデータが記録されていた領域を開放することができる。

【0104】

また、reference management atom内において、参照回数の情報を出すref#counterだけでなく、参照しているユーザプログラムのファイル名の情報を持たせてもよい。これによって、ユーザによってアフレコデータの削除が指示された場合に、該アフレコデータが他のユーザプログラムファイルから参照されているか否かの情報の他に、どのファイルから参照されているかをユーザに提示することが可能になる。

【0105】

さらに、各ユーザプログラムからアフレコデータのどの部分が参照されているかを示す参照範囲の情報を持たせることもできる。このような参照範囲情報により、アフレコデータ自体は複数のユーザプログラムから参照されていても要らなくなった部分のみを削除できるため、より有効にPRU領域を活用することができる。

【0106】

〔実施の形態2〕

次に、本発明の実施の形態2について説明する。実施の形態1と異なり、アフレコしたデータ毎にファイルを分けて管理する。尚、本実施の形態2は上述の実

施の形態1と類似するため、相違点に絞って説明する。

【0107】

先ず、アフレコデータが2種類あるときのディスク上の配置とファイルの関係を図25に示す。光ディスク106上の物理的な記録では、CU毎にアフレコデータブロックおよびオリジナルブロック（すなわちオリジナルオーディオブロックおよびオリジナルビデオブロック）が配置されており、アフレコデータブロックはアフレコデータ#Aを記録する領域、アフレコデータ#Bを記録する領域、未割り当て領域からなっている。そして、アフレコデータ#Aに係る各データはアフレコAファイルに、アフレコデータ#Bに係る各データはアフレコBファイルに、未割り当て領域はダミーファイルにて管理されている。

【0108】

<初期記録時の処理>

実施の形態2における初期記録時の処理は、実施の形態1と共通であるので、説明を省略する。

【0109】

<アフレコデータ記録時の処理>

実施の形態2におけるアフレコデータ記録時の処理を説明する。まず、アフレコデータを追加するユーザプログラムのQuickTime管理情報をRAM102に読み込み、オリジナルプログラムを特定してオリジナルプログラムのQuickTime管理情報をRAM102に読み込む。さらに、ファイルシステム管理情報をRAM102に読み込む。PRUの位置の特定とアフレコデータ記録領域判定は実施の形態1と同じであるので説明を省く。

【0110】

次に、アフレコデータをディスク上のPRU領域に記録し、新規にアフレコ用のファイルを作成してファイルシステム管理情報を更新する。このとき、新規に作成したアフレコデータのサイズに応じて未割り当て領域で構成されるダミーファイルのサイズを小さくし、ファイルシステム管理情報を更新する。

【0111】

次に、オリジナルプログラムのQuickTime管理情報に関して、アフレコデータ

用のトラックを新規に作成し、ユーザが指定した区間のsample tableを作成し、ref#counterフィールドに‘1’をセットする。また、data reference atomには先程作成したアフレコファイル名をセットする。

【0112】

次に、オリジナルプログラムの未割り当てトラックのsample tableのchunk offsetとsizeを、アフレコデータ記録後のPRUの残り領域に対応するよう更新する。

【0113】

次に、ユーザプログラム側の管理情報に、先程作成したオリジナルプログラムのアフレコトラックのsample tableをコピーする。

【0114】

最後に、RAM102上のオリジナルプログラムおよびユーザプログラムのQuickTime管理情報、及びファイルシステム管理情報を光ディスク106に記録する。

【0115】

別の種類のアフレコデータを追加するときの処理も上記と同様である。すなわちアフレコデータの種類の数だけファイルが作成される。

【0116】

<ユーザプログラム作成時の処理>

実施の形態2では、複数のアフレコデータを持つオリジナルプログラムの中からユーザプログラムが参照しているアフレコデータのトラックを特定する際、ユーザプログラムのアフレコトラックのData reference atomに書かれているファイル名と同じファイル名を参照しているトラックをオリジナルプログラムの中からサーチする。オリジナルプログラムは、ユーザプログラムで参照するオーディオビデオファイルから特定する。それ以外の処理は実施の形態1と共通である。

【0117】

<ユーザプログラム削除時の処理>

実施の形態2では、削除するユーザプログラムのアフレコデータが、オリジナルプログラムのどのトラックで管理されているかを特定する際、上記ユーザプロ

グラム作成時の処理で説明したのと同様に、data reference atomの参照ファイル名を用いて行う。

【0118】

また、ユーザプログラムを削除することによって、ref#counterが0になった場合、このアフレコデータをPRU領域から削除し、開放した領域を未割り当て領域に返却するが、QuickTime管理情報の書き換えに加え、該アフレコデータのファイルも削除し、未割り当て領域で構成されるダミーファイルのサイズを削除したファイルサイズに応じて大きくし、ファイルシステム管理情報の更新をする。それ以外の処理は実施の形態1と共通である。

【0119】

このように、実施の形態1では、アフレコファイルのサイズとアドレスは常に変わらないため、アフレコデータの削除が管理情報の書き換えのみで行うことができ、ファイルシステムの書き換えを必要としないものの、アフレコデータを特定する際に、サンプルのchunk offset値からアフレコファイル中の特定データを探索せねばならなかった。

【0120】

これに対し、本実施の形態2によると、アフレコデータ毎にファイルが分かれているため、data reference atomの参照ファイル名から一発でアフレコデータを特定することができ、データの特定がより簡便に、より迅速に行うことができる。ただし、管理情報の操作の他、ファイルシステムの操作も必要になる。

【0121】

<実施の形態2における変形例>

実施の形態2では、オリジナルプログラムの管理情報内で、アフレコデータの種類毎にトラックを分け、その中で参照回数を管理していたが、ディスク上に1個存在する管理情報ファイルであるインデックスファイル内にて管理する構成であっても良い。この場合、図26に示すように、ファイル名などのアフレコデータファイルを特定するための情報と参照回数情報を関連つけて管理する。

【0122】

尚、上記実施の形態1および2において、用いられる記録媒体は光ディスクと

したが、本発明を適用できる記録媒体は光ディスクに限定されるものではなく、ランダムアクセス可能な記録媒体（例えばハードディスク）であればよい。

【0123】

【発明の効果】

本発明のデータ記録方法は、以上のように、記録媒体上に、A Vデータの同一のシーンを使って複数のユーザプログラムを作成可能であり、かつ、アフレコデータが複数のユーザプログラムから参照可能であり、ユーザプログラムとアフレコデータとの参照関係を管理する管理情報を上記記録媒体上に記録する構成である。

【0124】

それゆえ、例えば、あるユーザプログラムを削除するにあたって、この削除されるユーザプログラムで参照されているアフレコデータを併せて削除しようとする場合には、削除されるべきアフレコデータは削除されるユーザプログラムのみで参照されるものであることを確認することが必要となるが、このような確認が上記管理情報によって容易に行えるといった効果を奏する。

【0125】

また、上記データ記録方法では、上記アフレコデータは、複数のユーザプログラムにおいて参照されるアフレコデータ群を含む領域中の部分領域であり、前記領域はひとつのアフレコファイルとして記録されており、上記管理情報は、オリジナルプログラムに対して1つ存在する管理情報ファイル中にアフレコデータ毎に記録される構成とすることができる。

【0126】

また、上記データ記録方法では、上記アフレコデータは、同一のユーザプログラムにおいて参照されるアフレコデータ群ごとにアフレコファイルとして記録されており、上記管理情報は、複数のオリジナルプログラムに対して1つ存在する管理情報ファイル中にアフレコファイル毎に記録される構成とすることができる。

【0127】

それゆえ、アフレコファイルのサイズとアドレスとが常に変わらないような管

理が可能であり、アフレコデータの削除が管理情報の書き換えのみで行うことができるため、ファイルシステムの書き換えを必要としないといった効果を奏する。

【0128】

また、上記データ記録方法では、上記アフレコデータは、同一のユーザプログラムにおいて参照されるアフレコデータ群ごとにアフレコファイルとして記録されており、上記管理情報は、オリジナルプログラムに対して1つ存在する管理情報ファイル中にアフレコファイル毎に記録される構成とすることができる。

【0129】

それゆえ、アフレコデータ毎にアフレコファイルが分かれているため、アフレコデータを特定する際、データのオフセットなどから検索しなくとも、各アフレコファイルの参照ファイル名から即座に特定できるため、データの特定がより簡便に、より迅速に行うことができるといった効果を奏する。

【0130】

また、上記データ記録方法では、上記管理情報は、アフレコデータが複数のユーザプログラムから参照されている場合における参照回数を含む構成とすることができる。

【0131】

それゆえ、例えば、あるユーザプログラムを削除するにあたって、上記管理情報に含まれる参照回数を確認することによって、削除されるユーザプログラムで参照されているアフレコデータが、この削除されるユーザプログラムのみで参照されるものであるか否かの判断が容易に行えるといった効果を奏する。

【0132】

また、上記データ記録方法では、上記管理情報は、アフレコデータを参照しているユーザプログラム名を含む構成とすることができる。

【0133】

それゆえ、あるユーザプログラムを削除するにあたって、この削除されるユーザプログラムで参照されているアフレコデータが他のユーザプログラムファイルから参照されているか否かの情報の他に、どのファイルから参照されているかを

ユーザに提示することができるといった効果を奏する。

【0134】

また、上記データ記録方法では、上記管理情報は、ユーザプログラムによって参照されているアフレコデータの参照範囲の情報を含む構成とすることができる。

【0135】

それゆえ、各ユーザプログラムからアフレコデータのどの部分が参照されているかを示す参照範囲の情報を持たせることで、アフレコデータ自体は複数のユーザプログラムから参照されていても要らなくなった部分のみを削除できるため、より有効に記録媒体の記録領域を活用することができるといった効果を奏する。

【図面の簡単な説明】

【図1】

本発明の一実施形態を示すものであり、アフレコデータ記録時の処理の流れを示したフローチャートである。

【図2】

本発明に係るデータ記録装置の概略構成を示すブロック図である。

【図3】

図3 (a) ~ (c) は、QuickTimeファイルフォーマットにおける管理情報とAVストリームとの関係を示す図である。

【図4】

QuickTimeファイルフォーマットにおけるMovie atomの概要を示す図である。

【図5】

QuickTimeファイルフォーマットにおけるTrack atomの概要を示す図である。

【図6】

QuickTimeファイルフォーマットにおけるTrack header atomの構成を示す図である。

【図7】

QuickTimeファイルフォーマットにおけるMedia atomの構成を示す図である。

【図8】

QuickTimeファイルフォーマットにおけるMedia information atomの構成を示す図である。

【図 9】

Sample table atomによるデータ管理の例を示す図である。

【図 10】

QuickTimeファイルフォーマットにおけるSample table atomの構成を示す図である。

【図 11】

図 11 (a) はQuickTimeファイルフォーマットにおけるSample description atomの構成を示す図であり、図 11 (b) はSound Sample description Tableの構成を示す図である。

【図 12】

QuickTimeファイルフォーマットにおけるEdit atomの構成を示す図である。

【図 13】

図 13 (a) ~ (c) は、Edit atomによる再生範囲指定の例を示す図である。

【図 14】

QuickTimeファイルフォーマットにおけるUser data atomの構成を示す図である。

【図 15】

A Vストリームの構成を示す図である。

【図 16】

図 16 (a) , (b) は、UDFにおける管理情報の関係を示す図である。

【図 17】

本発明の第 1 の実施例における、QuickTimeファイルフォーマットにおけるUser data atomの構成を示す図である。

【図 18】

本発明の実施の形態 1 における、記録直後のオリジナルプログラムの管理情報を示す図である。

【図 19】

図 19 (a), (b) は、本発明の実施の形態 1 における、ユーザプログラム作成時の未割り当てトラックの変更を示す図である。

【図 20】

本発明の実施の形態 1 における、ディスク上でのデータ配置とファイルとの関係を示した図である。

【図 21】

本発明の実施の形態 1 における、ユーザプログラムの編集の例を示した図である。

【図 22】

本発明の実施の形態 1 における、ユーザプログラム削除の処理の流れを示したフローチャートである。

【図 23】

本発明の実施の形態 1 における、ユーザプログラムの削除直前の参照関係を示した図である。

【図 24】

本発明の実施の形態 1 における、ユーザプログラムの削除直後の参照関係を示した図である。

【図 25】

本発明の実施の形態 2 における、ディスク上でのデータの配置とファイルの関係を示す図である。

【図 26】

本発明の実施の形態 2 における、ユーザプログラムの参照関係とインデックスファイルとの関係を示した図である。

【図 27】

従来技術におけるディスク上での記録形態を示す図である。

【符号の説明】

100 バス

101 ホスト CPU

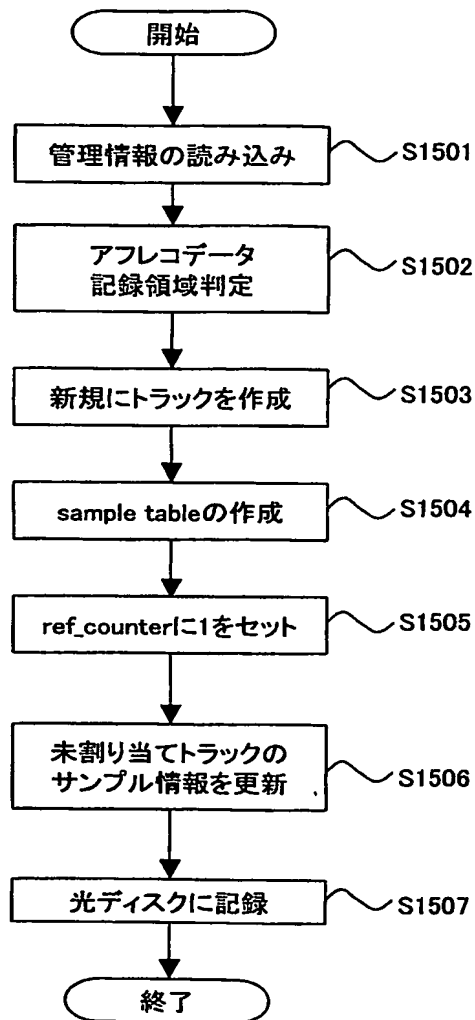
102 RAM

106 光ディスク（記録媒体）

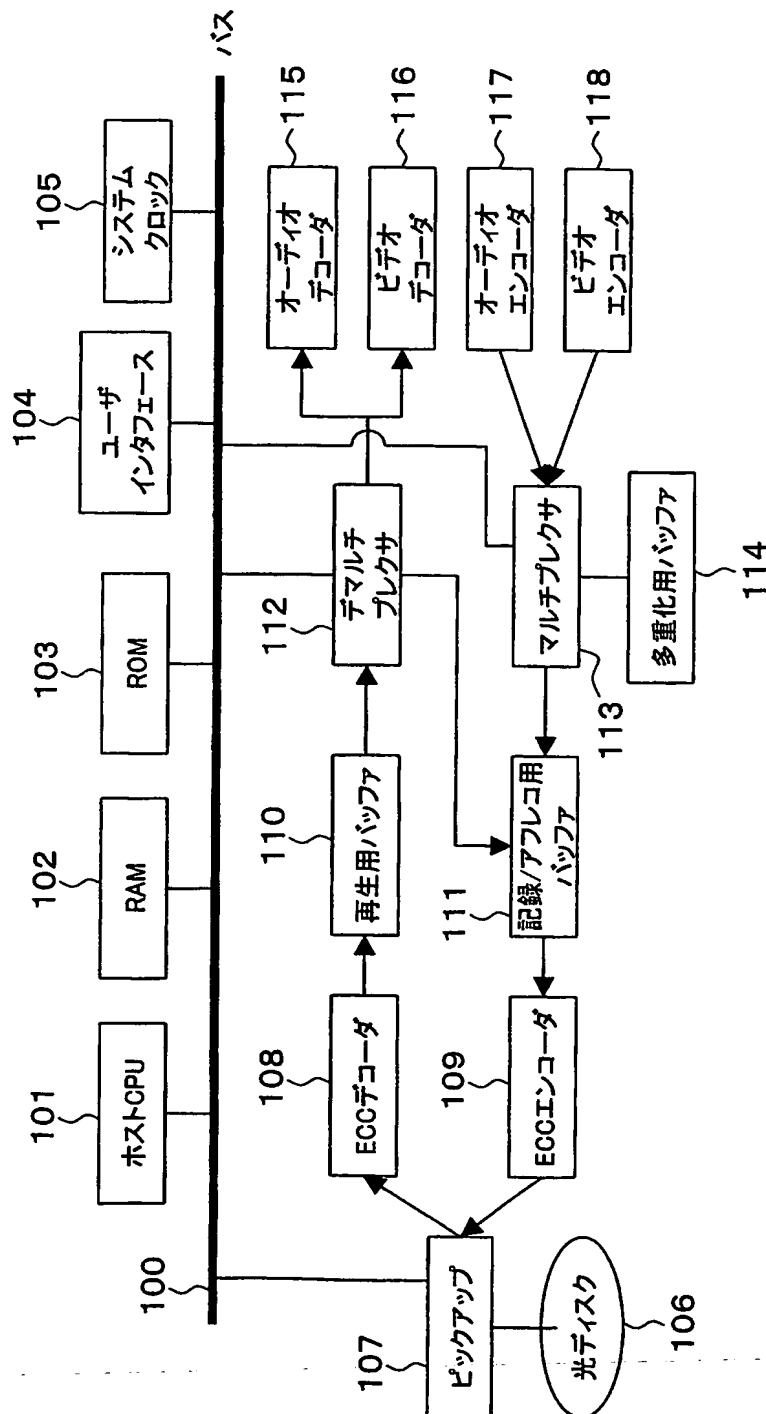
107 ピックアップ

【書類名】 図面

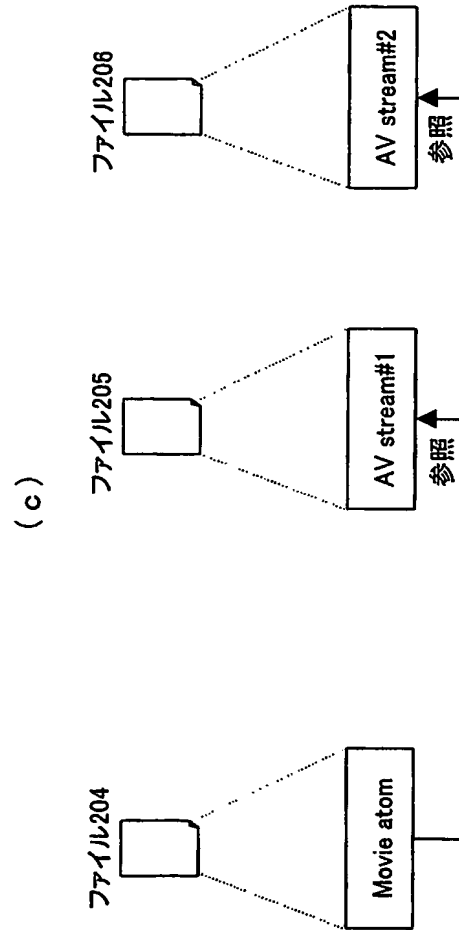
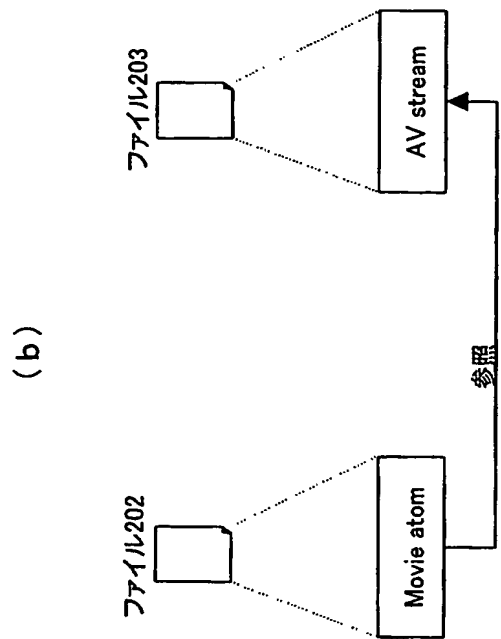
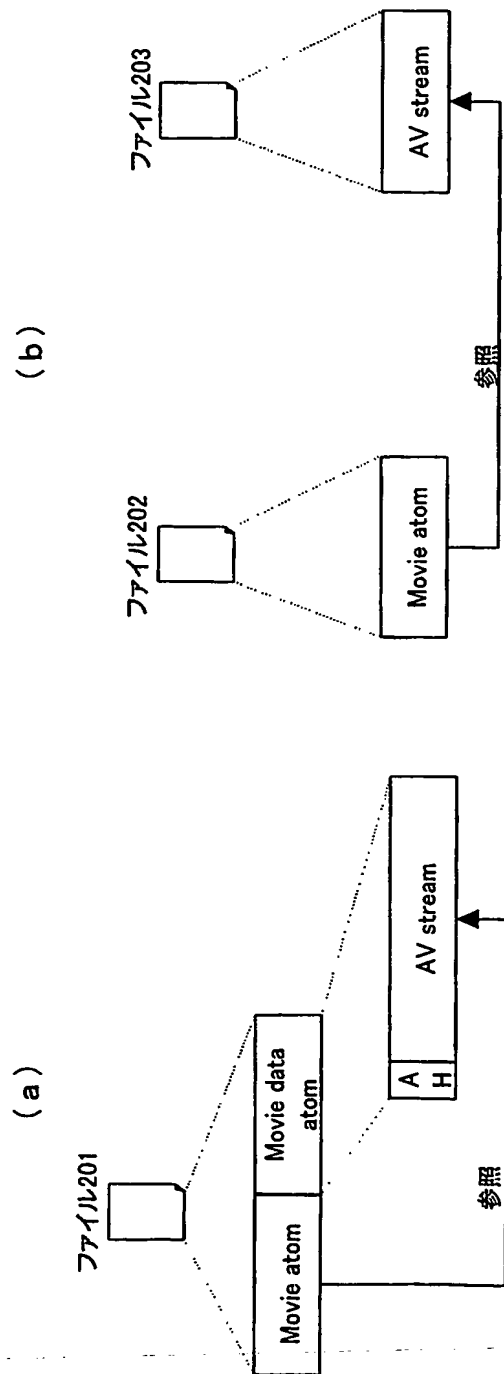
【図 1】



【図 2】



【図 3】



【図 4】

```
Movie atom {  
    Atom size  
    Type(='moov')  
    Movie header atom  
    Track atom (video track)  
    Track atom (main audio track)  
    :  
    User data atom  
}
```

【図 5】

```
Track atom {  
    Atom size  
    Type(='trak')  
    Track header atom  
    Edit atom  
    Track reference atom  
    Media atom  
    User data atom  
    :  
}
```


【図 6】

```
Track header atom {
    Atom size
    Type(='tkhd')
    Version
    Flags
    Creation time
    Modification time
    Track ID
    Reserved
    Duration
    Reserved
    Layer
    Alternate group
    Volume
    Reserved
    Matrix structure
    Track width
    Track height
}
```

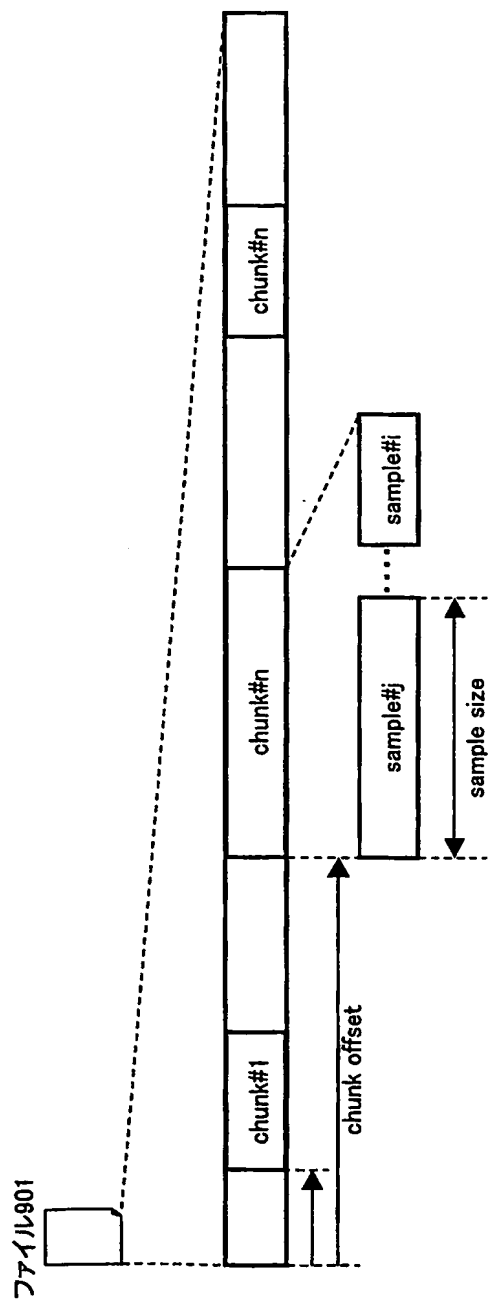
【図 7】

```
Media atom {
    Atom size
    Type(='mdia')
    Media header atom
    Handler reference atom
    Media information atom
    User data atom
    :
}
```

【図 8】

```
Media information atom {  
  Atom size  
  Type(='minf')  
  {Video or Sound or Base} media information header atom  
  Handler reference atom  
  Data information atom  
  Sample table atom  
}
```

【図 9】



【図 10】

```
Sample table atom {  
    Atom size  
    Type(='stbl')  
    Sample description atom  
    Time-to-sample atom  
    Sync sample atom  
    Sample-to-chunk atom  
    Sample size atom  
    Chunk offset atom  
}
```

【図 11】

```
(a)  sample description atom {  
      Atom size  
      Type(='std')  
      version  
      flags  
      number-of-entries  
      for(i=0; i<number-of-entries; ++){  
        Sample Description Table  
      }  
}  
  
(b)  sound sample description Table {  
      size  
      data-format  
      reserved  
      data-reference-index  
      //sample description fields for sound media//  
      version  
      :  
}
```

【図 12】

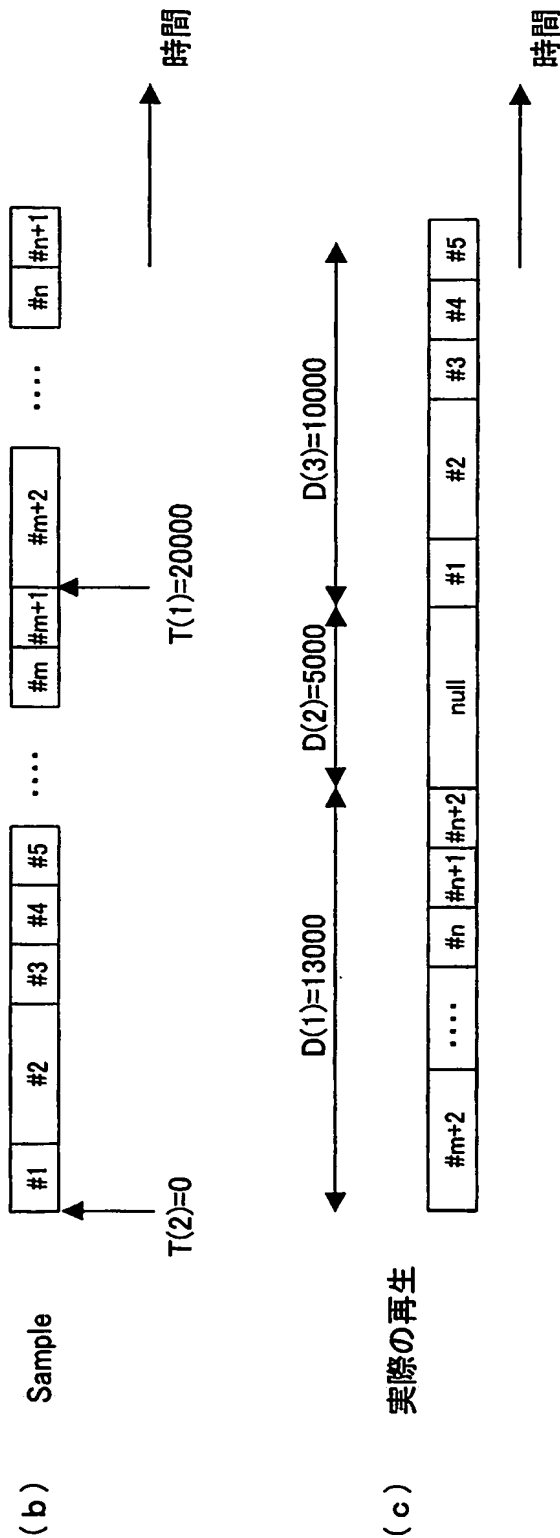
```
Edit atom {
    Atom size
    Type(='edts')
    Edit list atom
}

Edit list atom {
    Atom size
    Type(='elst')
    Versions
    Flags
    Number of entries(=N)
    for (i = 0; i < N; i++){
        Track duration
        Media time
        Media rate
    }
}
```

【図 13】

Entry Number	Track duration D(i)	Media time T(i)	Media rate R(i)
#1	13000	20000	1
#2	5000	-1	1
#3	10000	0	1

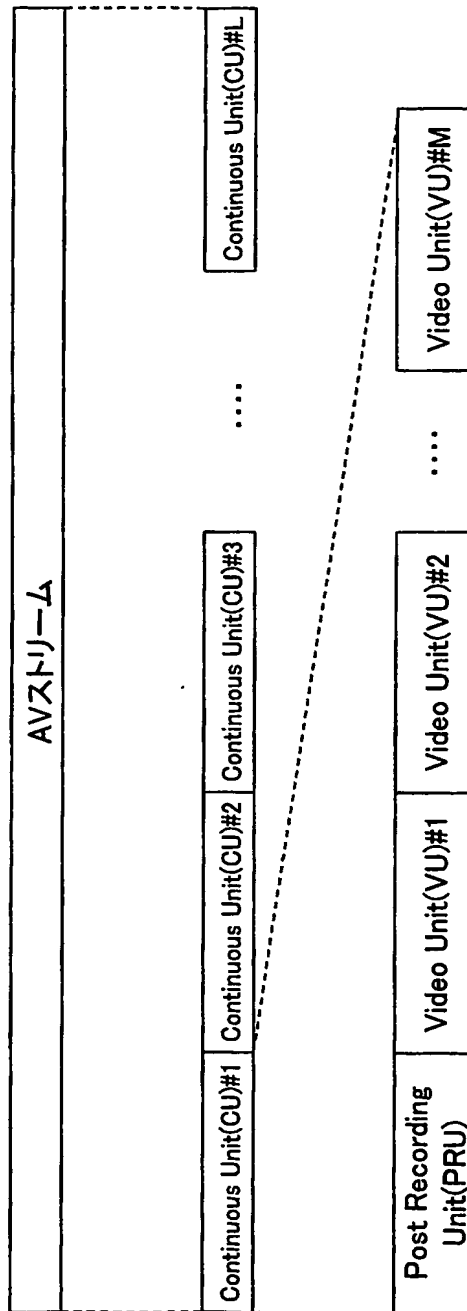
(a)



【図 14】

```
User data atom {  
    Atom size  
    Type(='udta')  
    for (i=0;i<N; i++){  
        Atom size  
        Type  
        User data  
    }  
}
```

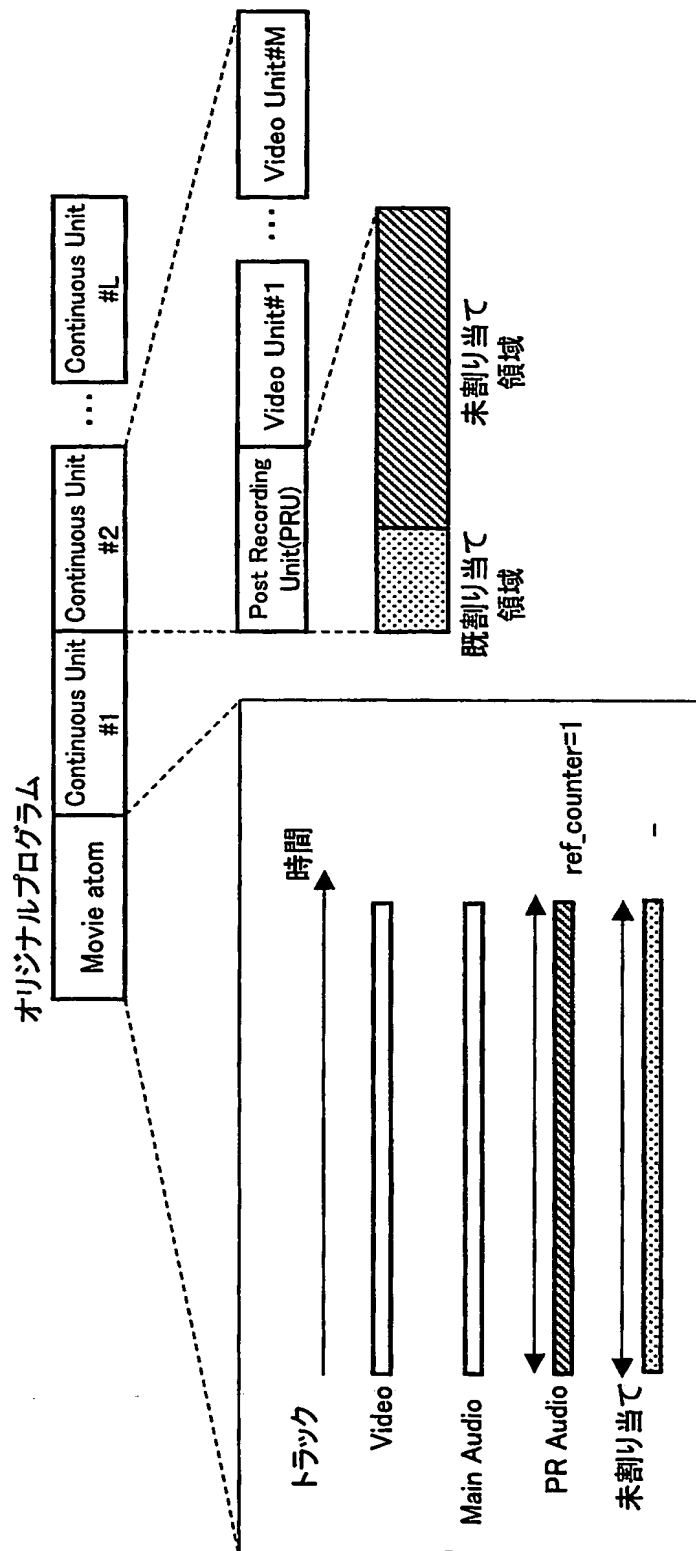

【図 15】



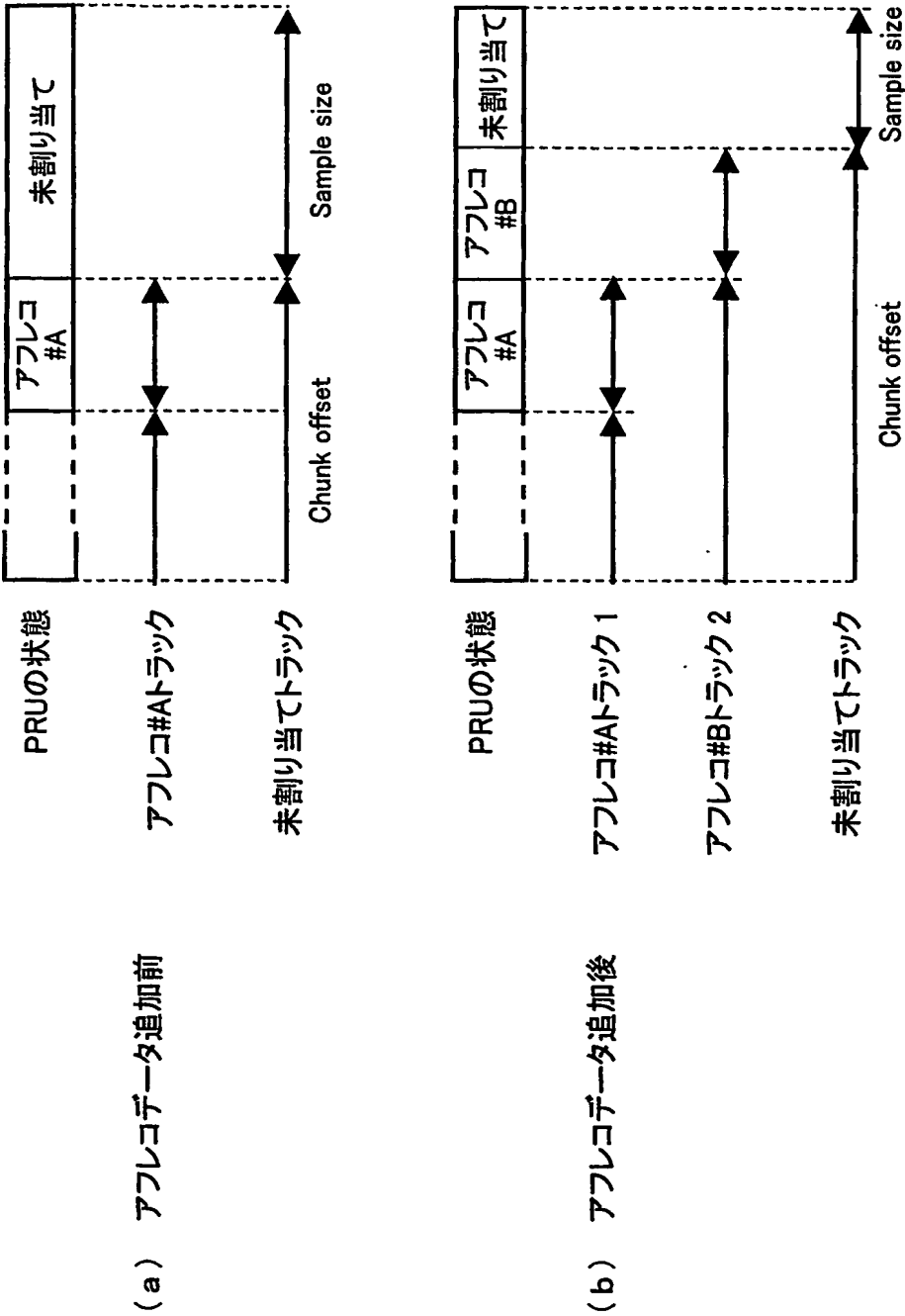
【図 17】

```
User data atom {  
    Atom size  
    Type(='udta')  
    for (i=0; i<N; i++){  
        Atom size  
        Type(='rfmg':reference management)  
        ref_counter  
    }  
}
```

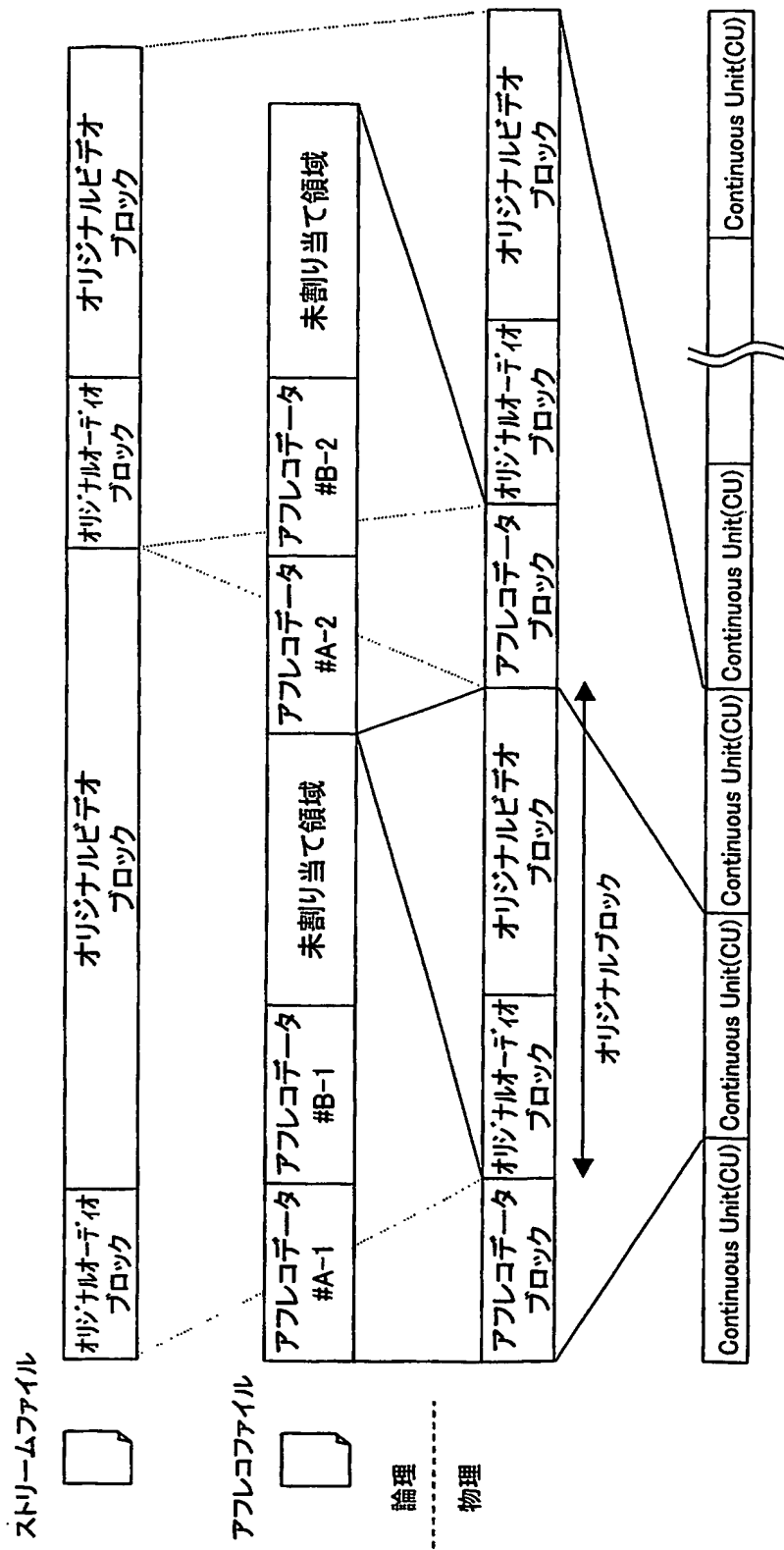
【図 18】



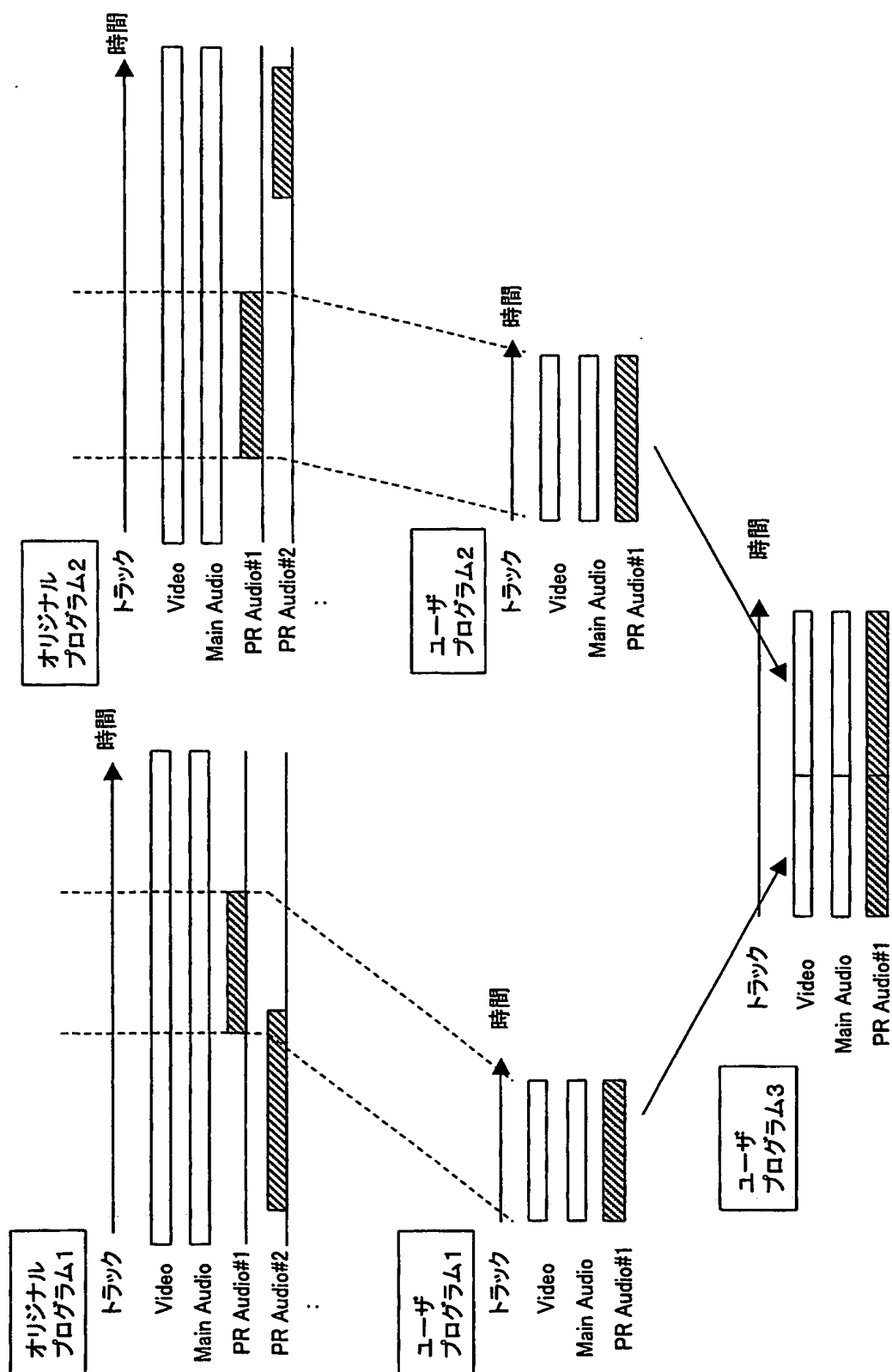
【図 19】



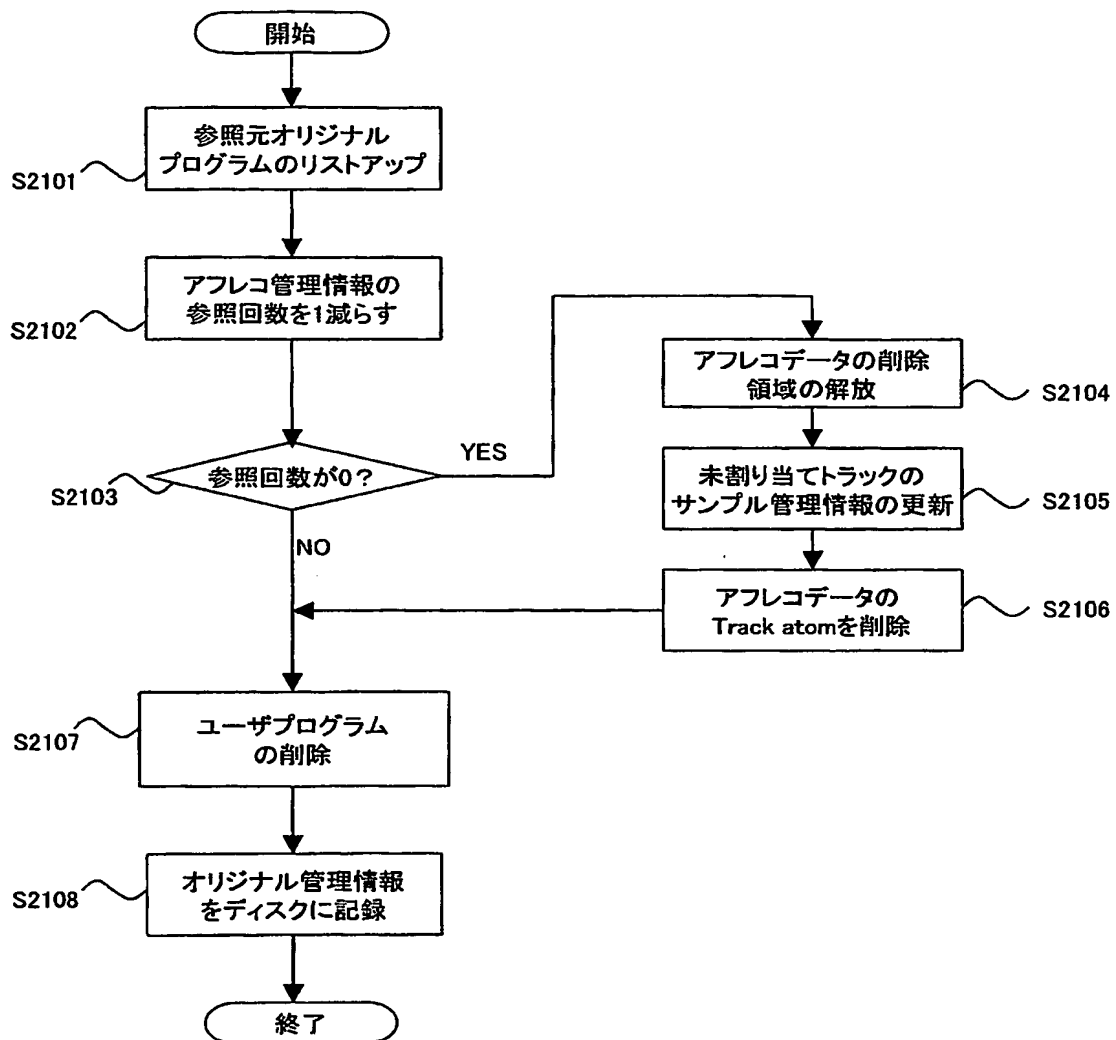
【図 20】



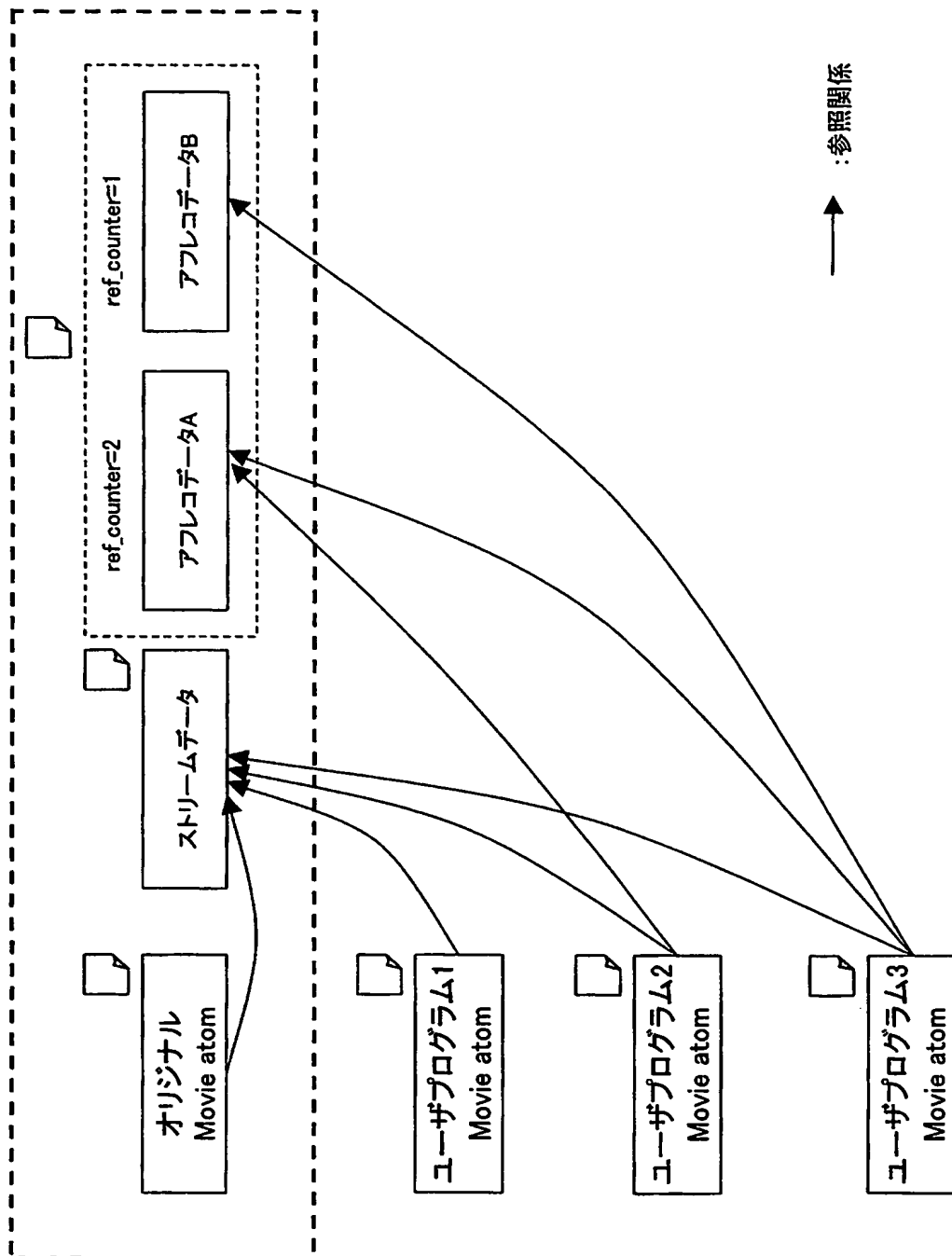
【図 21】



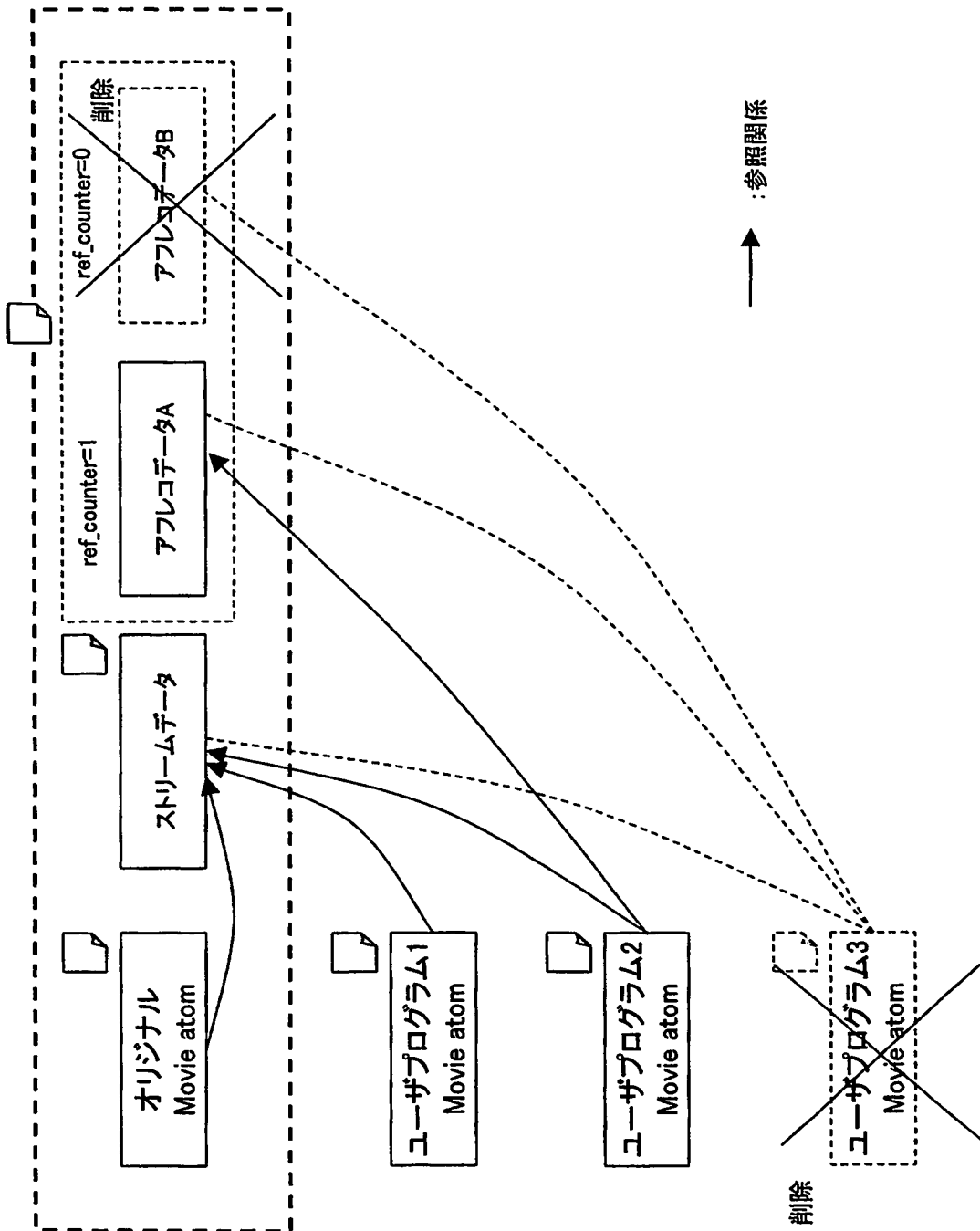
【図 22】



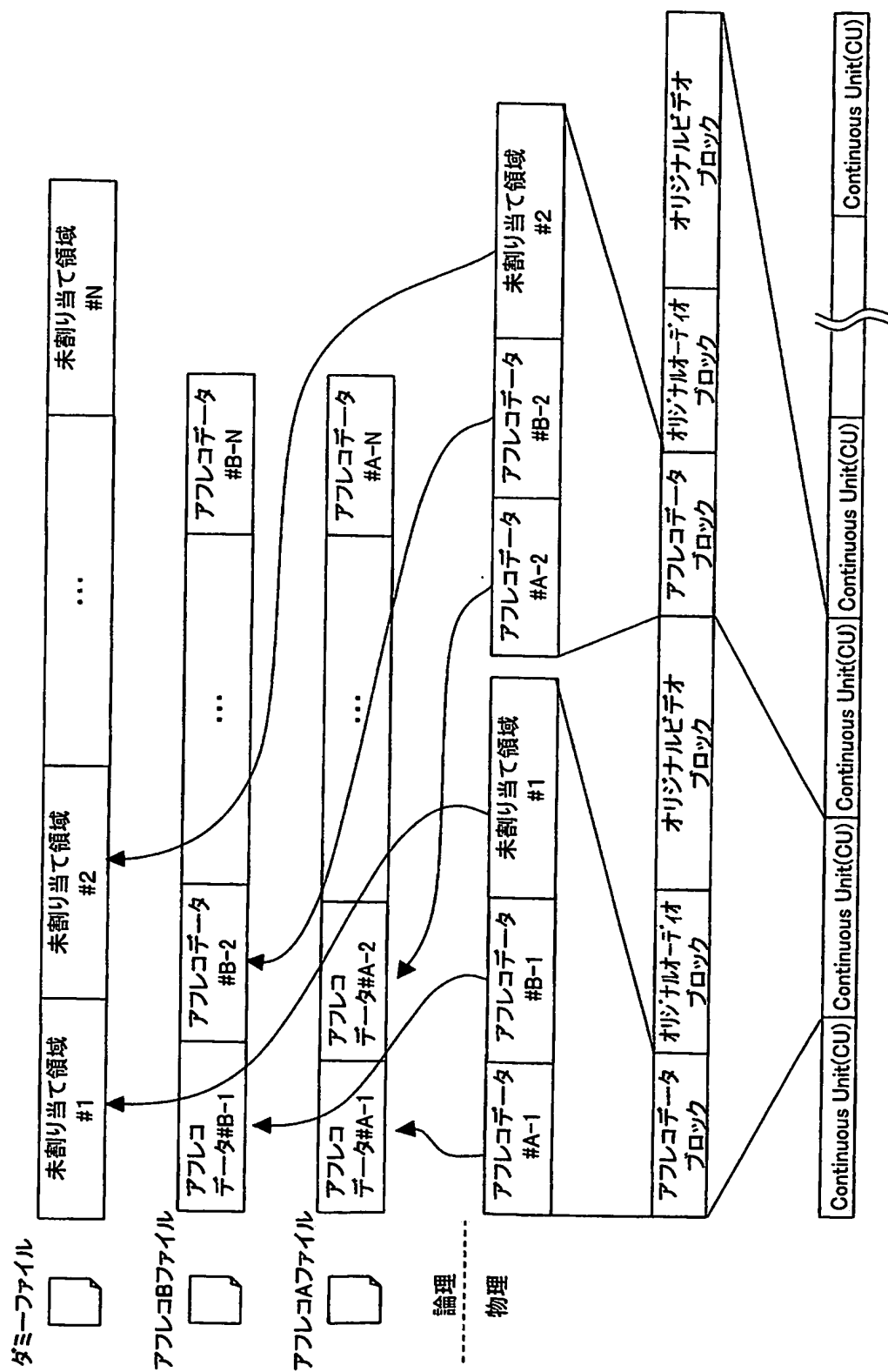
【図 23】



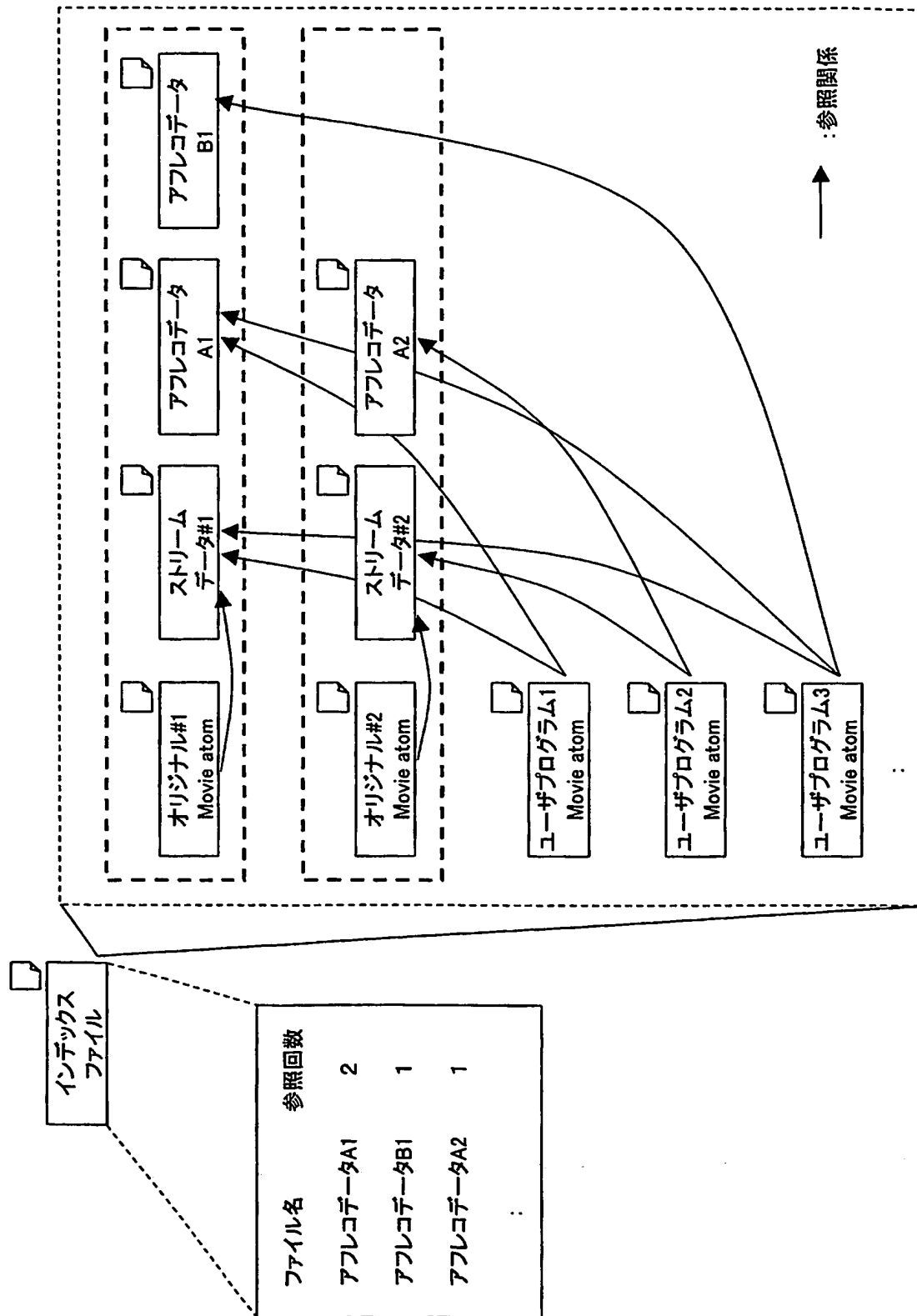
【図 24】



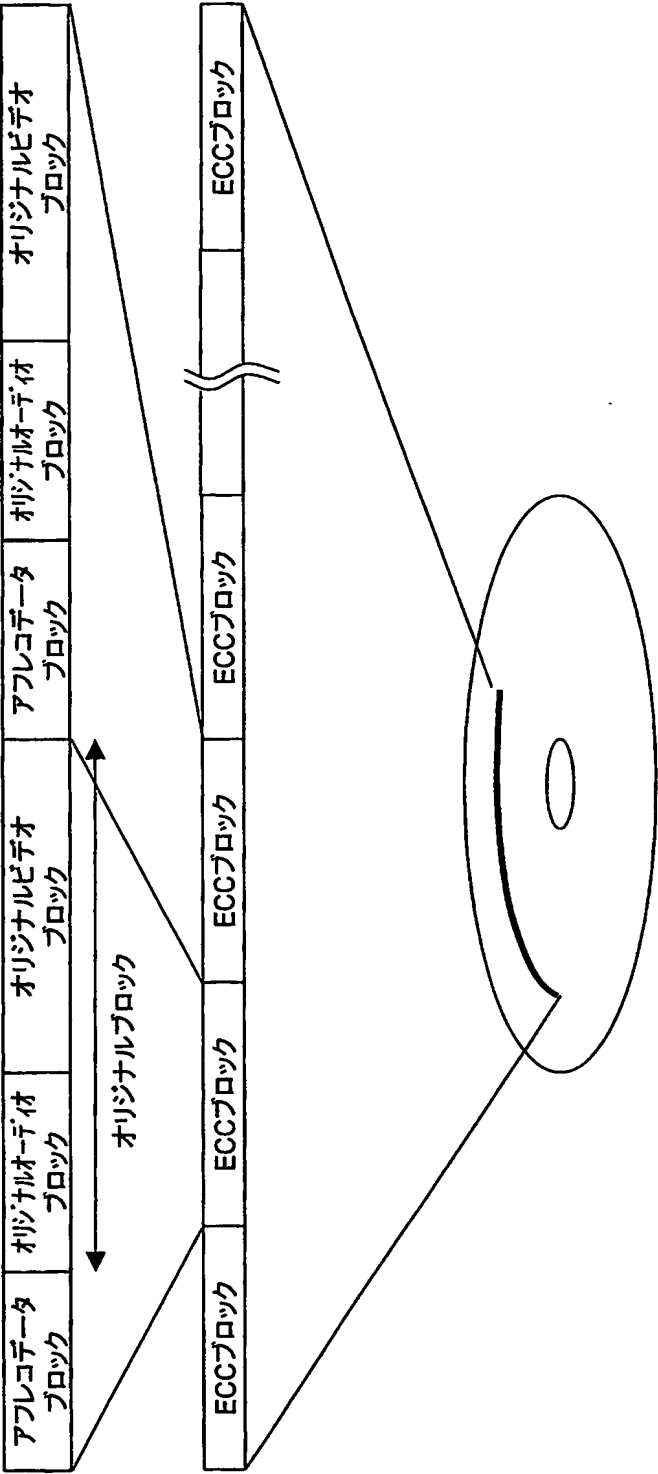
【図25】



【図 26】



【図 27】



【書類名】 要約書

【要約】

【課題】 ユーザプログラムの削除時、該ユーザプログラムにおいて参照していたアフレコデータが削除可能であるか否かの判断を容易にする。

【解決手段】 オリジナルプログラムを外部参照したユーザプログラムを作成し、それにアフレコデータを記録するにあたって、アフレコデータのユーザプログラムからの参照回数を示すref#counterを設定する（S1505）。設定されたref#counterは、作成されたユーザプログラムと共に光ディスクに記録される（S1507）。

【選択図】 図1

特願 2 0 0 3 - 1 6 2 9 2 8

出 願 人 履 歴 情 報

識別番号 [0 0 0 0 0 5 0 4 9]

1. 変更年月日 1 9 9 0 年 8 月 2 9 日

[変更理由] 新規登録

住 所 大阪府大阪市阿倍野区長池町 2 2 番 2 2 号

氏 名 シャープ株式会社